S-100
$2.00
U.S.A.
MICROSYSTEMS™
SEPT/OCT 1980 VOL. 1/NO. 5

HANDLING CP/M UTILITIES, BIOS & FILE OPERATIONS

See Pages 10-32

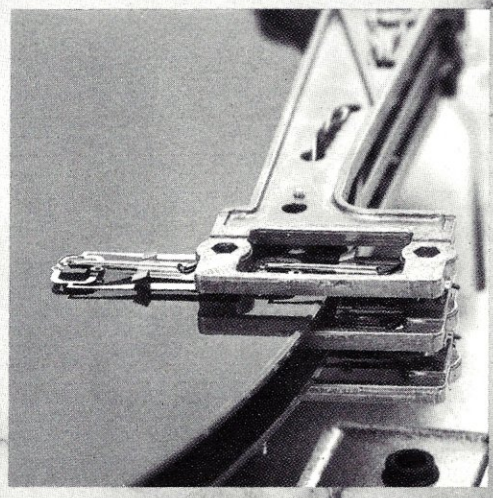
Also in this Issue

and more

Complete Table of Contents on Page 3

26 MEGABYTES
$4995.

DRIVE A HARD BARGAIN!



Suddenly, S-100 microcomputer systems can easily handle 100 million bytes. Because Morrow Designs™ now offers the first 26 megabyte hard disk memory for S-100 systems—the DISCUS M26™ Hard Disk System.

It has 26 megabytes of useable memory (29 megabytes unformatted). And it's expandable to 104 megabytes.

The DISCUS M26™ system is delivered complete—a 26 megabyte hard disk drive, controller, cables and operating system—for just $4995. Up to three additional drives can be added, $4495 apiece.

The DISCUS M26™ system features the Shugart SA4008 Winchester-type sealed media hard disk drive, in a handsome metal cabinet with fan and power supply.

The single-board S-100 controller incorporates intelligence to supervise all data transfers, communicating with the CPU via three I/O ports (command, status, and data). The controller has the ability to generate interrupts at the completion of each command to increase system throughput. There is a 512 byte sector buffer on-board. And each sector can be individually write-protected for data base security.

The operating system furnished with DISCUS M26™ systems is the widely accepted CP/M* 2.0.

See the biggest, most cost-efficient memory ever introduced for S-100 systems, now at your local computer shop. If unavailable locally, write Morrow Designs,™ 5221 Central Avenue, Richmond, CA 94804. Or call (415) 524-2101, weekdays 10-5 Pacific Time.

*CP/M is a trademark of Digital Research.

MORROW DESIGNS™

Thinker Toys™

NEW! TPM* for TRS-80 Model II
NEW! System/6 Package
Computer Design Labs

Z80* Disk Software

We have acquired the rights to all TDL software (& hardware). TDL software has long had the reputation of being the best in the industry. Computer Design Labs will continue to maintain, evolve and add to this superior line of quality software.
— Carl Galletti and Roger Amidon, owners.

Software with Manual/Manual Alone

All of the software below is available on any of the following media for operation with a Z80 CPU using the CP/M* or similar type disk operating system (such as our own TPM*).

for TRS-80* CP/M (Model I or II)
for 8" CP/M (soft sectored single density)
for 5¼" CP/M (soft sectored single density)
for 5¼" North Star CP/M (single density)
for 5¼" North Star CP/M (double density)

BASIC I

A powerful and fast Z80 Basic interpreter with EDIT, RENUMBER, TRACE, PRINT USING, assembly language subroutine CALL, LOADGO for "chaining", COPY to move text, EXCHANGE, KILL, LINE INPUT, error intercept, sequential file handling in both ASCII and binary formats, and much, much more. It runs in a little over 12 K. An excellent choice for games since the precision was limited to 7 digits in order to make it one of the fastest around. $49.95/$15.

BASIC II

Basic I but with 12 digit precision to make its power available to the business world with only a slight sacrifice in speed. Still runs faster than most other Basics (even those with much less precision). $99.95/$15.

BUSINESS BASIC

The most powerful Basic for business applications. It adds to Basic II with random or sequential disk files in either fixed or variable record lengths, simultaneous access to multiple disk files, PRIVACY command to prohibit user access to source code, global editing, added math functions, and disk file maintenance capability without leaving Basic (list, rename, or delete). $179.95/$25.

ZEDIT

A character oriented text editor with 26 commands and "macro" capability for stringing multiple commands together. Included are a complete array of character move, add, delete, and display function. $49.95./$15.

ZTEL

Z80 Text Editing Language - Not just a text editor. Actually a language which allows you to edit text and also write, save, and recall programs which manipulate text. Commands include conditional branching, subroutine calls, iteration, block move, expression evaluation, and much more. Contains 36 value registers and 10 text registers. Be creative! Manipulate text with commands you write using Ztel. $79.95/$25.

TOP

A Z80 Text Output Processor which will do text formatting for manuals, documents, and other word processing jobs. Works with any text editor. Does justification, page numbering and headings, spacing, centering, and much more! $79.95/$25.

MACRO I

A macro assembler which will generate relocateable or absolute code for the 8080 or Z80 using standard Intel mnemonics plus TDL/Z80 extensions. Functions include 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical operations, local and global symbols, chaining files, linking capability with optional linker, and recursive/reiterative macros. This assembler is so powerful you'll think it is doing all the work for you. It actually makes assembly language programming much less of an effort and more creative. $79.95/$20.

MACRO II

Expands upon Macro I's linking capability (which is useful but somewhat limited) thereby being able to take full advantage of the optional Linker. Also a time and date function has been added and the listing capability improved. $99.95/$25.

LINKER

How many times have you written the same subroutine in each new program? Top notch professional programmers compile a library of these subroutines and use a Linker to tie them together at assembly time. Development time is thus drastically reduced and becomes comparable to writing in a high level language but with all the speed of assembly language. So, get the new CDL Linker and start writing programs in a fraction of the time it took before. Linker is compatible with Macro I & II as well as TDL/Xitan assemblers version 2.0 or later. $79.95/$20.

DEBUG I

Many programmers give up on writing in assembly language even though they know their programs would be faster and more powerful. To them assembly language seems difficult to understand and follow, as well as being a nightmare to debug. Well, not with proper tools like Debug I. With Debug I you can easily follow the flow of any Z80 or 8080 program. Trace the program one step at a time or 10 steps or whatever you like. At each step you will be able to see the instruction executed and what it did. If desired, modifications can then be made before continuing. It's all under your control. You can even skip displaying a subroutine call and up to seven breakpoints can be set during execution. Use of Debug I can pay for itself many times over by saving you valuable debugging time. $79.95/$20.

DEBUG II

This is an expanded debugger which has all of the features of Debug I plus many more. You can "trap" (i.e. trace a program until a set of register, flag, and/or memory conditions occur). Also, instructions may be entered and executed immediately. This makes it easy to learn new instructions by examining registers/memory before and after. And a RADIX function allows changing between ASCII, binary, decimal, hex, octal, signed decimal, or split octal. All these features and more add up to give you a very powerful development tool. Both Debug I and II must run on a Z80 but will debug both Z80 and 8080 code. $99.95/$20.

ZAPPLE

A Z80 executive and debug monitor. Capable of search, ASCII put and display, read and write to I/O ports, hex math, breakpoint, execute, move, fill, display, read and write in Intel or binary format tape, and more! on disk $34.95/$15.

APPLE

8080 version of Zapple $34.95/$15.

NEW! TPM now available for TRS-80 Model II!

TPM*

A NEW Z80 disk operation system! This is not CP/M*. It's better! You can still run any program which runs with CP/M* but unlike CP/M* this operating system was written specifically for the Z80* and takes full advantage of its extra powerful instruction set. In other words its not warmed over 8080 code! Available for TRS-80* (Model I or II). Tarbell, Xitan DDDC, SD Sales "VERSAFLOPPY", North Star (SD&DD), and Digital (Micro) Systems. $79.95/$25.

SYSTEM MONITOR BOARD (SMB II)

A complete I/O board for S-100 systems. 2 serial ports, 2 parallel ports, 1200/2400 baud cassette tape interface, sockets for 2K of RAM, 3-2708/2716 EPROM's or ROM, jump on reset circuitry. Bare board $49.95/$20.

ROM FOR SMB II

2KX8 masked ROM of Zapple monitor. Includes source listing $34.95/$15.

PAYROLL (source code only)

The Osborne package. Requires C Basic 2.
5" disks $124.95 (manual not included)
8" disks $ 99.95 (manual not included)
Manual $20.00

ACCOUNTS PAYABLE/RECEIVABLE (source code only)

By Osborne, Requires C Basic 2
5" disks $124.95 (manual not included)
8" $99.95 (manual not included)
Manual $20.00

GENERAL LEDGER (source code only)

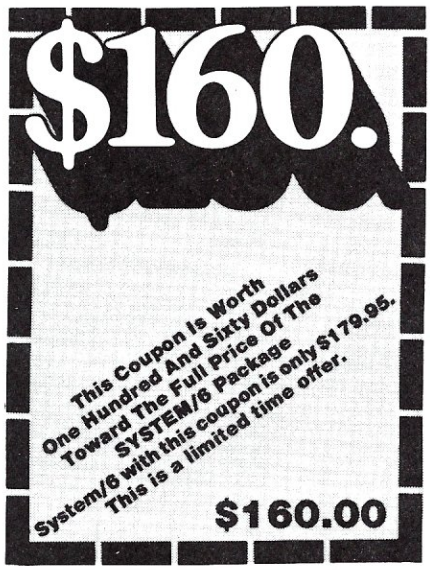
By Osborne. Requires C Basic 2
5" disks $99.95 (manual not included)
8" disks $99.95 (manual not included)
Manual $20.00

C BASIC 2

Required for Osborne software. $99.95/$20.

SYSTEM/6

TPM with utilities, Basic I interpreter, Basic E compiler, Macro I assembler, Debug I debugger, and ZEDIT text editor.
Above purchased separately costs $339.75
Special introductory offer: Only $179.75 with coupon!!



ORDERING INFORMATION

Visa, Master Charge and C.O.D. O.K. To order call or write with the following information.

1. Name of Product (e.g. Macro I)
2. Media (e.g. 8" CP/M)
3. Price and method of payment (e.g. C.O.D.) include credit card info. if applicable.
4. Name, Address and Phone number.
5. For TPM orders only: Indicate if for TRS 80, Tarbell, Xitan DDDC, SD Sales (5¼" or 8"). ICOM (5¼" or 8"), North Star (single or double density) or Digital (Micro) Systems.
6. N.J. residents add 5% sales tax.

Manual cost applicable against price of subsequent software purchase in any item except for the Osborne software.

For information and tech queries call
609-599-2146

For phone orders ONLY call toll free
1-800-327-9191
Ext. 676
(Except Florida)

OEMS

Many CDL products are available for licensing to OEMs. Write to Carl Galletti with your requirements.

* Z80 is a trademark of Zilog
* TRS-80 is a trademark for Radio Shack
* TPM is a trademark of Computer Design Labs. It is not CP/M*
* CP/M is a trademark of Digital Research

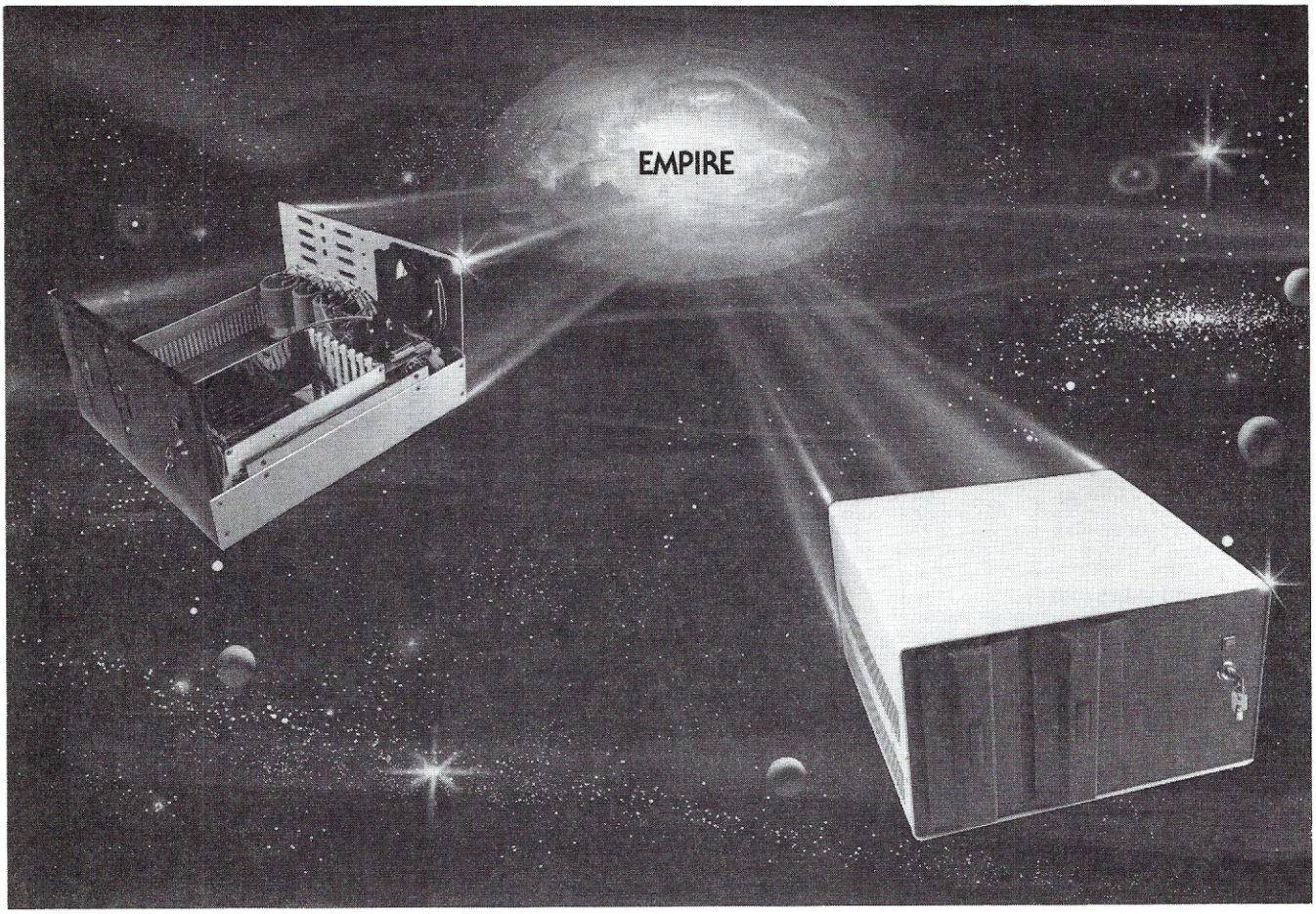
Prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.

COMPUTER DESIGN LABS

342 Columbus Avenue
Trenton, N.J. 08629

The Empire has expanded!

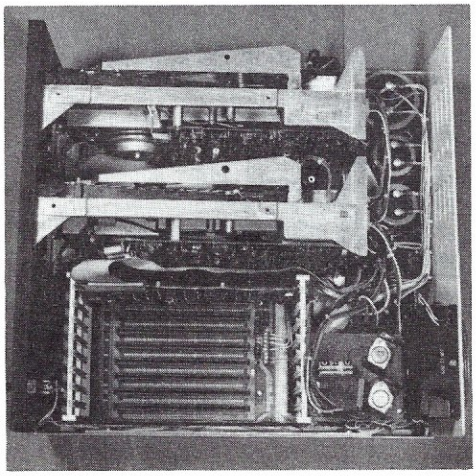


New Mainframe opens more areas for development

In one quantum leap Tarbell has expanded its popular Empire (the vertical disk subsystem) into a full line. This entire series now encompasses 5 variations. Each one contains different components so the S-100 system designer, hobbyist, or serious business user can arrive at the exact custom state he wants and needs.

The basic Empire still includes two Shugart or Siemens 8" disk drives; the compact cabinet with fan and power supply; a Tarbell floppy disk interface; CP/M*; Tarbell BASIC; the necessary cables, connectors and complete documentation. Naturally, it's fully assembled and Tarbell tested.

The new, top of the line Empire contains the basic model's components with the Tarbell design-approved Mainframe. Beside the 8-slot S-100 motherboard with an active terminated bus, there's a cardcage with card guides and a double-density interface.



You're the master of your Empire
You can call the shots in the Empire. Tarbell's made sure of that by offering them as complete subsystem packages . . . or, as separate units. For example, the mainframe may be ordered with 1, 2 or no drives. Whichever way you go, however, you always get the reliability of Tarbell tested components and leadership-engineering.

To get control of your own Empire, see your quality computer store for quick delivery. Or, contact us for dealer locations or further information.

CP/M is a trademark of Digital Research.

Tarbell Electronics

950 Dovlen Place - Suite B
Carson, California, 90746
(213) 538-4251 / 538-2254

S-100
MICROSYSTEMS™

VOLUME 1 NUMBER 5

SEPTEMBER/OCTOBER 1980

Editorial Correspondence should be sent to: S-100 MICROSYSTEMS, BOX 1192, Mountainside, NJ 07092.

S-100 MICROSYSTEMS is seeking articles on S-100 software, hardware and applications. Program listings should be typed on white paper with a new ribbon. Articles should be typed 40 characters/inch at 10 pitch. Author's name, address and phone number should be included on first page of article and all pages should be numbered. Photos are desirable and should be black and white glossy.

Commercial advertising is welcomed. Write to S-100 MICROSYSTEMS, 93 Washington St., Morristown, NJ 07960, or phone Claudette Moore at 201-267-4558.

*TMK Digital Research

IN THIS ISSUE

DEPARTMENTS

S-100 MICROSYSTEMS (USPS 529-530) is published bi-monthly by Libes, Inc., 995 Chimney Ridge, Springfield, NJ 07801. Subscription (USA) is $9.50 per year. Controlled circulation paid at Baltimore, Maryland.

POSTMASTER: Send address changes to S-100 MICROSYSTEMS, PO Box 789-M, Morristown, NJ 07960.

The Editor's Page

by Sol Libes

The past year has seen very sizeable increases in the prices of software packages. The next few months should see sizeable increases in the prices for microcomputer software. This is all due to significant changes in the software marketplace.

I remember only five years ago buying my first copy of Basic for only $5. It was a mini-version of about 2K bytes of code on paper tape. It came in the mail in a standard number ten size envelope (I remember the postage was only 10 cents then) and included 5 photocopied pages of instructions. The author sold it directly, as a hobby. He had a good full time job and this was just a little project, on the side. The program provided only the very fundamental Basic functions. Storing a Basic program, by today's standards, was a real hassle and there were a few bugs in the program. But at $5, who could complain?

Today, things have changed radically. I have changed...from a pure hobbyist-to an entrepreneur that relys on his system for income. My needs have changed. I need powerful software with a lot of bells and whistles. Further, I need software which will allow me to do my tasks quickly. And also, I need software that is bug free and does not consume my time with debugging.

Today I think little of spending $100 for a software package, and I have spent as much as $500 for a package. I think that I am typical of most microcomputer users. I am sure that most of the computer hobbyists of two to five years ago are the entrepreneurs of today. It seems like all of my old computer hobbyist buddies are making money with their systems, one way or another. Although we still play around with our systems just for the fun of it, most of our time is now spent on income producing projects.

The microcomputer market has thus been shifting from what was originally strictly a hobby to one that is now predominantly oriented to professional users. The likelihood is that the coming year will see much more of this, with business applications finally coming to dominate the microcomputer area.

The microcomputer software supplier area was started by part-timers working in their basements and attics.....like my Basic supplier. There are still plenty of them around in fact, their number seems to be increasing. However, the software marketplace has matured greatly. The new breed of software supplier is catering to the professional and business users. Customers are not price sensitive. Rather, they are willing and able to pay high prices but are demanding value for their money. They want sophisticated software. They want a lot of support and handholding. They want software that is bug free and delivers on promises. This demand for higher quality software requires that the software supplier be a professionally run organization that is there to stay. This costs money and hence the increases in software prices reflect this.

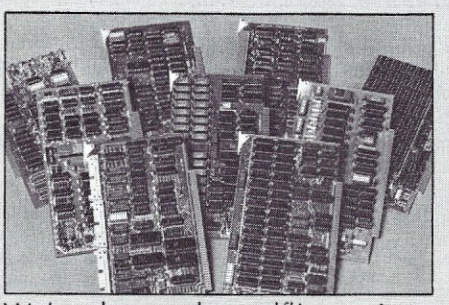
An example of the changes in software prices is reflected by the price of the VisiCalc software package sold by Personal Software. This very popular software package was first introduced with a suggested selling price of $100. It was sold through dealers and hence the company may have netted only $60-70 on each sale. Probably, most of this amount was eaten up by production and marketing costs leaving little, if anything, to cover development and customer support costs. The result was that Personal Software recently increased the price to $200—a 100% increase—and the likelihood is that the price will be increased further. Most software suppliers, however, are attempting to hold prices on previously released packages and are raising prices on new releases.

There are already a few microcomputer software packages selling in the $1,000 area. These are the business-oriented packages. It is likely that by the end of next year typical software package costs will have shifted from the present $100-$500 to the $5000-$1000 area, with few going to the $2000 area. After all, keep in mind that software packages for mini and large size computer systems sell in the over $10,000 area with several going for over $50,000.

One distressing fact is that although software suppliers have been raising software prices, the level of customer support and pre-testing of software has still not changed as dramatically. □

At Intersystems, "dump" is an instruction. Not a way of life.

(Or, when you're ready for IEEE S-100, will your computer be ready for you?)



We're about to be gadflies again.

While everyone's been busy trying to convince you that large buses housed in strong metal boxes will guarantee versatility and ward off obsolescence, we've been busy with something better. Solving the *real* problem with the first line of computer products *built from the ground up to conform to the new IEEE S-100 Bus Standard*. Offering you extra versatility in 8-bit applications today. And a full 16 bits tomorrow.

We call our new line Series II.™ And even if you don't need the full 24-bit address for up to 16 megabytes (!) of memory right now, they're something to think about. Because of all the performance, flexibility and economy they offer. Whether you're looking at a new mainframe, expanding your present one or upgrading your system with an eye to the future. (Series II boards are compatible with most existing S-100 systems and *all* IEEE S-100 Standard cards as other manufacturers get around to building them.)

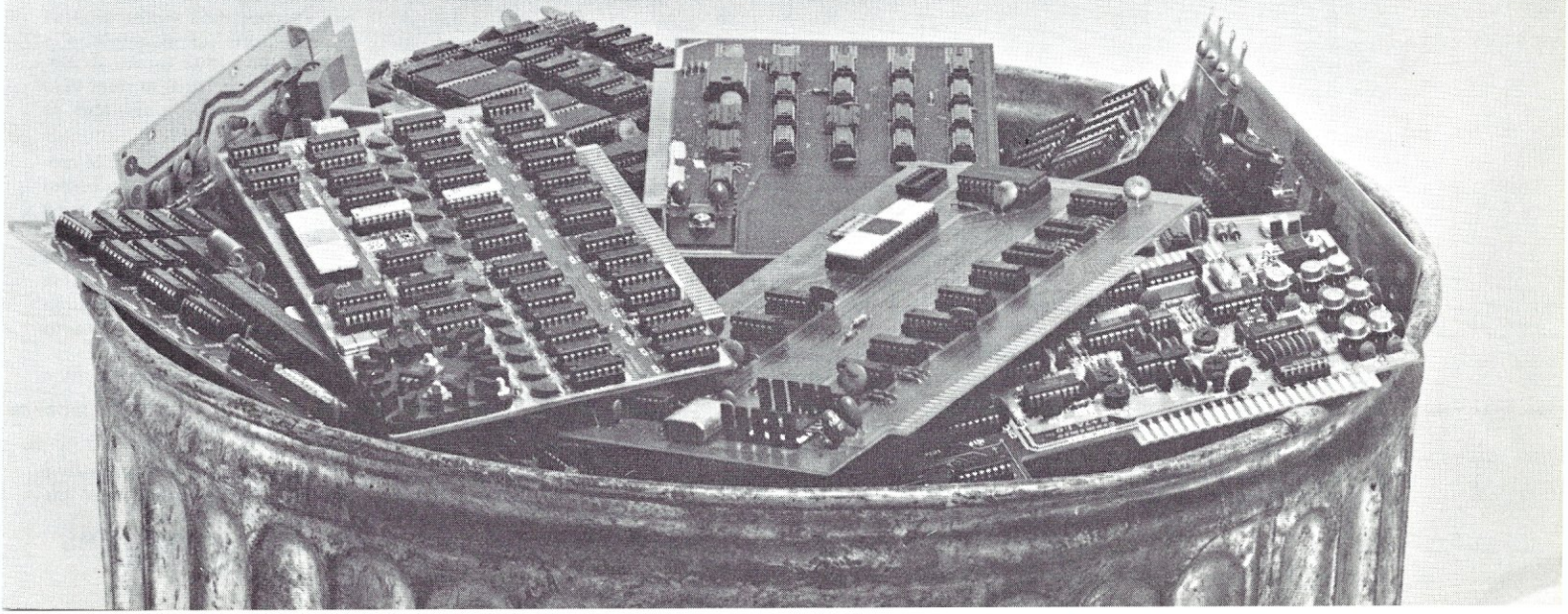
Consider some of the features: Reliable operation to 4MHz and beyond. Full compatibility with 8- and 16-bit CPUs, peripherals and other devices. *Eight* levels of prioritized interrupts. Up to 16 individually-addressable DMA devices, with IEEE Standard overlapped operation. User-selectable functions addressed by DIP-switch or jumpers, eliminating soldering. And that's just for openers.

The best part is that all this heady stuff is available *now*! In our advanced processor—a full IEEE Bus Master featuring Memory Map™ addressing to a full megabyte. Our fast, flexible 16K Static RAM and 64K Dynamic RAM boards. An incredibly versatile and economical 2-serial, 4-parallel Multiple I/O board. 8-bit A/D-D/A converter. Our Double-Density High-Speed Disk Controller. And what is undoubtedly the most flexible front panel in the business. Everything you need for a complete IEEE S-100 system. Available separately, or all together in our new DPS-1 Mainframe!

Whatever your needs, why dump your money into obsolete products labelled "IEEE timing compatible" or other words people use to make up for a lack of product. See the future now, at your Intersystems dealer or call/write for our new catalog. We'll tell you all about Series II and the new IEEE S-100 Bus we helped pioneer. Because it doesn't make sense to buy yesterday's products when tomorrow's are already here.

IntersSystems™

Ithaca Intersystems Inc.,
1650 Hanshaw Road/P.O. Box 91,
Ithaca, NY 14850
607-257-0190/TWX: 510 255 4346



LETTERS TO THE EDITOR

Dear Editor:

I'm glad that there is finally someone competent supporting the S-100 bus. The magazines have been doing at best a mediocre job and its been getting progressively worse and worse. The S-100 bus certainly has had its problems and I'm well aware of them with my collection of over 150 S-100 boards. However, there's a lot to be said in favor of the S-100 bus and that just has not been done properly by the traditional magazines. The creativeness, innovativeness, and imagination of many of the S-100 designers has me in absolute awe. It really irks me when someone says an S-100 machine is no good because its a hobbyist machine and is therefore of inferior quality. True, there is a lot of schlock in the S-100 area, but that's because there has been so much designed for it and because of the hesitancy of the traditional magazines to print legitimate criticisms of poorly designed and obsolete products. There is more depth in the S-100 area than anywhere else. Unfortunately, some of the really first rate quality products in the S-100 area (such as the Computalker, Biotech products, ADS Noisemaker, Speechlab, Digital Graphic Systems CAT-100, Cambridge Development Labs graphic boards, Casheab music synthesizers, Marinchip 9900 MPU, Sublogic software, etc.) have not gotten the exposure and attention they deserve from the traditional magazines even though they've been around for several years. And as a result, many of these fine products are about to die.

One area of great hope and promise for the S-100 bus is the 16-bit microcomputer. Unfortunately, despite the IEEE S-100 standards, this could be a nightmare for the unwary consumer. For example, some 16-bit processors will work with only 16-bit memory and not the 8-bit memory that most of us S-100 users presently own. The new Ithaca Intersystems Z8000, the re-designed Alpha Micro 100T, and the TECHMAR 8086 are this way. Also it turns out that it is optional in the IEEE S-100 standards whether a manufacturer who designs IEEE S-100 memory boards wants to support a 16-bit data path for his memory. Most of the IEEE S-100 memory products on the market right now such as Godbout, Thilnker Toys, Measurement System & Control, etc. does not support true 16-bit data transfers. I should also add that though some of the new 16-bit products are electrically IEEE S-100 compatible, they may not be physically IEEE S-100 compatible. For example, the new Alpha-Micro 100T and Piceon's 32K 16-bit word memory cards are both 10 inches tall!!! This means that you cannot fit them into a normal IMSAI, ALTAIR, TEI, Vector Graphics, Godbout, or North Star chassis. Then there's Marinchip's 32K 16-bit word memory card which will only work with their 16-bit computer and will not work with any of the 8-bit computers. Also some of the memory-mapped devices such as the North Star minifloppy interface, the North Star hardware multiply/divide board, and Thinkertoy's floppy disk probably won't work with all of the new 16-bitters. Since many of these new 16-bit computers can directly address 128K bytes of memory or more, consumers should begin to consider seriously forcing the manufacturers to provide memory with error-correcting code capability, as is done on mini-computers and traditional mainframes. And I haven't even mentioned anything in the area of software yet! Its pretty obvious that something needs to be done and said in the magazines to provide a smoother transition for S-100 consumers into the world of the powerful 16-bitters if the S-100 bus is to be successful in this area. I want it to be.

Kenneth Young
Los Angelos, CA

Dear Editor:

The following might be an item of interest to some of your readers: I have found a paperback book called "8080/8085 Software Design" to be an excellent publication. Perhaps it might even be called a reference book, as it contains a library of commonly used 8080 assembly language routines, which saves the reader considerable effort in not having to 'roll his own', and also, probably saves him from doing many of these in the worst possible manner.

Much of today's documentation, I find, is poorly written; and I think of it as not unlike asking a native for directions to get somewhere. Invariably the reply is something like, "Follow the road you're on all the way down. You'll see it. You can't possibly miss it." Having traveled up and down that same road each day, the native soon becomes oblivious to the fact that there may be three forks in the road along the way. To him, his way home, is the only 'straight road'. A newcomer usually can describe a route better than someone presenting himself as an authority.

In any event, Dr. Christopher Titus seems to have an ability to point out the 'forks' in the 'road'. Mine was the Sams version consisting of two volumes at $10. each, although I believe that E&L instruments has since come out with the same text in one volume, also $10., which would seem to be a better buy.

Ernest J. Zitke
Brockton, Mass.

Dear Editor:

I am currently making a CPU selection for my S-100 system, and would like to advise you of some discrepancies I have found in the S-100 Processor Boards and Manufacturers article published on page 50 of the Jul-Aug issue of S-100 MICROSYSTEMS

1) I ordered a Z-80 processor board from Digital Research Computers in Texas. My check was returned with a note saying that they no longer offered the board that I had ordered. In fact, their literature accompanying my returned check makes no mention of any processor board now being offered.

2) The article mentions the 8085 and/or 8088 boards (Compupro) from Godbout Electronics, but fails to mention their enhanced Z-80A board. This was advertised on the back cover of the July/August issue of S-100 MICROSYSTEMS.

3) Page 202 of the July 1980 issue of KILOBAUD advertises a 68000 board for the S-100 bus with on board Z-80 emulation. It is to be marketed by Vandata, 17541 Stone Ave. N., Seattle WA 98133 PH (206)-542-8370. I have learned through a telephone conversation with an employee at Vandata that a prototype of this board is functioning and is being built by Imperical Research Group (IRG) in Kent WA. And that it may be available as early as October 1980.

Best of luck with S-100 Microsystems. I am finding it a useful and enjoyable tool while putting together my system. Perhaps when its all working I will be able to share in an article, some facet of this system's operation or construction as it relates to the S-100 bus.

Tim Hamilton
Creswell, OR

Dear Editor:

Many reasons for the relative success of systems like the

Apple and the TRS 80 have been given. Mass production by a large well known company is not the only reason, as evidenced by the popularity of the Apple. Much of the innovative and useful software written for these products would be nothing without the graphics.

In my opinion, one of the main reasons for this success, a reason which should be considered carefully by the S-100 community, is our lack of a standard video graphics system. Hardware variety is both the joy and the curse of the S-100 world. According to my count, there are at least 21 different manufacturers of S-100 video boards alone, with some firms offering up to 3 different models of video boards. This does not count the wide variety of terminals. Some of the boards and terminals have no graphics capability except what can be managed with text characters. Others sport a wide variety of graphics formats, from character block graphics characters up to HiRes boards programmable down to the last pixel. This very variety virtually guarantees that writing software for S-100 graphics will rarely be very profitable.

Fortunately, due to bus, processor, exchange, and text screen standards, there is a very large amount of non-graphics oriented software around, much of it business and utility programs.

With a new generation of HiRes Color Video boards appearing, how can we encourage programmers to use more graphics in their programs, not just in games, but in business software also? I can see three concievable avenues: (1) unlikely; vote on a hardware standard video system which every one interested would but (this would be unfair to many companies). (2) Get the companies to agree to at least one standard—each board would be switchable to run in this mode (this would require modifications to many existing boards and some could not be modified at all). (3) Cooperate in publishing complete tables of operating parameters for existing boards and systems, and prescriptions for translating formats where possible.

All of these ideas have serious drawbacks, but they are a start. I strongly urge you to make MICROSYSTEMS a coordinating medium for attacking this problem.

For example: can readers agree on specifications for a desireable video board like the following:

1) true memory mapped, switchable location or phantom: 32K at 4MHz.
2) 24 by 80 with optional 48-51 by 80 Black & white & Gray & Color Capability.
4) both character block and pixel mode graphics, **ram** chr. generator.
5) Variety of graphics resolutions, but at least 512 by 512.
6) On-board keyboard port (parallel).
7) 256 character codes and inverse (each mode).
8) light pen hardware and software.
9) contiguous character blocks and decenders for text mode.
10) IEEE compatable.
11) software support especially utilities, CPM compatable.

Some of the parameters which should be included in comparison tables should be:

Company, Address, Tel.#
Name of Board, Accesories
A&T Cost
Color?
Pixel Matrix Size: Character Blocks
Memory Mapped?
Onboard Processor?
Max Graphic Resolution
Video Input or Output Type
Maximum Memory Size
Calling Address STD Utilities
Max Power Used (Amps)
Mem Protect Field
Cursor Control?
Max Char Stored (Test)
Rom or Ram Char. Generator
Video Ram Starting Address
Grey Scale? # of Colors
Terminal Emulator Mode
Max Bus Speed
Occupied Memory Areas
Max Char. Avail in Single Mode
Control Char. Map - Effects

I have a partial list of my own, but most of it is gleaned from advertisements. What we really need is information on each board from a user or user manual. A standard form could be printed in the magazine then owners of boards could fill in this hard to get information without having to write an article.

Another similar problem area is system memory maps. A standard one sheet memory map could be agreed on, with color coding indicating **Ram, Rom**, areas for Higher languages, Video **Ram**, Disk Operating systems, etc. Then software incompatabilities could be seen instantly and decisions made more rapidly. Maps of popular systems could be published in the magazine along with articles covering the featured system.

I hope you can use some of these ideas. I wish you and your venture the very best of luck.

John K. Strickland, Jr.
Leander, TX

Prices reflect distribution on 8" single density diskettes. If a format is requested which requires additional diskettes, a surcharge of $8. per additional diskette will be added. A surcharge of $25 will be added for software on CSSN format DC 300XL cartridges. Media charge for 5440 disk is $100

No.16: Take a byte.

Genuine CP/M for Apple II Available now!

All Lifeboat programs require CP/M, unless otherwise stated.
Software for most popular 8080/Z80 computer disk systems

Software with Manual / Manual Alone

☐ **CP/M* FLOPPY DISK OPERATING SYSTEM**—Digital Research's operating system configured for many popular micro-computers and disk systems:

System	Version	Price
New Apple II*	2.x	350/25 ◇
SoftCard* with Z80 Microsoft BASIC version 5 with high resolution graphics		
North Star Single Density	1.4	145/25 Ⓥ
North Star Double Density	1.4	145/25
North Star Single Density	2.x	170/25
North Star Double/Quad	2.x	170/25
Durango F-85	2.x	170/25
iCOM Micro-Disk 2411	1.4	145/25
iCOM 3712	1.4	170/25 Ⓥ *
iCOM 3812	1.4	170/25 *
Mits 3202/Altair 8800	1.4	145/25
Heath H8 + H17	1.4	145/25 Ⓜ
Heath H89	1.4	145/25 Ⓜ
Heath H89 by Magnolia	1.4	250/25 ◇
Heath H89 by Magnolia	2.x	300/25 ◇
Onyx C8001	2.x	300/25 ◇
Ohio Scientific C3	2.x	200/25
TRS-80 Model I	1.4	145/25 Ⓜ
TRS-80 Model II	2.x	170/25
TRS-80 Model II + Corvus	2.x	250/25
Processor Technology Helios II	1.4	145/25
Cromemco System 3	1.4	145/25
Intel MDS Single Density	1.4	145/25
Intel MDS Single Density	2.x	170/25
Micropolis Mod I	1.4	145/25 Ⓥ
Micropolis Mod II	1.4	145/25 Ⓥ

The following configurations are scheduled for release soon:

North Star Double/Quad + Corvus	2.x	250/25
North Star Horizon HD-1	2.x	250/25
Ohio Scientific C3-C	2.x	250/25
Micropolis Mod II	2.x	200/25
Mostek MDX STD Bus System	2.x	350/25 **
iCOM 3812	2.x	225/25 *
iCOM 4511/Pertec D3000	2.x	375/25 *+

*Software consists of the operating system, text editor, assembler, debugger and other utilities for file management and system maintenance. Complete set of Digital Research's documentation and additional implementation notes included. Systems marked * and ** include firmware on 2708 and 2716. Systems marked + include 5440 media charge. Systems marked Ⓜ require the special Ⓜ versions of software in this catalog. Systems marked Ⓥ have minor variants available to suit console interface of system. Call or write for full list of options. ◇ includes hardware addition to allow our standard versions of software to run under it.*

☐ **Z80 DEVELOPMENT PACKAGE**—Consists Ⓜ of: (1) disk file line editor, with global inter and ② intra-line facilities; (2) Z80 relocating assembler, Zilog/Mostek mnemonics, conditional assembly and cross reference table capabilities; (3) linking loader producing absolute Intel hex disk file ..... **$95/$20**

☐ **ZDT**—Z80 Monitor Debugger to break and Ⓜ examine registers with standard Zilog/ Ⓩ Mostek mnemonic disassembly displays. $35 when ordered with Z80 Development Package ..... **$50/$10**

AVOCET SYSTEMS

☐ **XASM-68**—Non-macro cross-assembler with nested conditionals and full range of pseudo operations. Assembles from standard Motorola MC6800 mnemonics to Intel hex ... **$200/$25**

☐ **XASM-65**—As XASM-68 for MOS Technology MCS-6500 series mnemonics ..... **$200/$25**

☐ **XASM-48**—As XASM-68 for Intel MCS-48 and UPI-41 families ..... **$200/$25**

☐ **XASM-18**—As XASM-68 for RCA 1802 ..... **$200/$25**

☐ **DISTEL**—Disk based disassembler to Intel 8080 or TDL/Xitan Z80 source code, listing and cross reference files, Intel or TDL/Xitan pseudo ops optional. Runs on 8080 ..... **$65/$10**

☐ **DISILOG**—As DISTEL to Zilog/Mostek Ⓜ mnemonic files ..... **$65/$10** ②

Software with Manual / Manual Alone

☐ **SMAL/80** Structured Macro Assembler ⓧ Language—Package of powerful general purpose text macro processor and SMAL structured language compiler. SMAL is an assembler language with IF-THEN-ELSE, LOOP-REPEAT-WHILE, DO-END, BEGIN-END constructs ..... **$75/$15**

PHOENIX SOFTWARE ASSOCIATES

☐ **PASM***—Z80 macro assembler, Intel/TDL ② mnemonics. Generates Intel hex format or relocatable code in either TDL Object Module format or PSA Relocatable Binary Module format. Supports text insertion, conditional branching within macros, recursive macro calls and parameter passing. ..... **$129/$25**

☐ **EDIT**—Character oriented text file editor. In- ② cludes macro definition capabilities. Handles insertion, deletion, searching, block move, etc. for files of any length. Does not require a CRT. ..... **$129/$25**

☐ **PLINK***—Two pass **disk-to-disk linkage edi- ② tor/loader** which can produce re-entrant, ROMable code. Can link programs that are larger than available memory for execution targeted on another machine. Full library capabilities. Input can be PSA Relocatable Binary Module, TDL Object Module or **Microsoft REL files**. Output can be a COM file, Intel hex file, TDL Object Module or PSA Relocatable file. ..... **$129/$25**

☐ **BUG* and μBUG***—Z80 interactive machine ② level debugging tools for program development. BUG has full symbolic trace and interactive assembly (mnemonics compatible with PASM). Dynamic breakpoints and conditional traps while tracing (even through ROM!). μBUG is a subset of BUG and is used in memory limited situations ..... **$129/$25**

DIGITAL RESEARCH

☐ **MP/M**—Installed for single density MDS-800. Multi-processing derivative of the CP/M operating system. Manual includes CP/M2 documentation ..... **$300/$50**

☐ **MAC**-8080—Macro assembler. Full Intel Ⓢ macro definitions. Pseudo Ops include RPC, IRP, REPT, TITLE, PAGE, and MACLIB. Produces absolute hex output plus symbol table file for use by SID and ZSID (see below) **$120/$15**

☐ **SID**-8080—Symbolic debugger. Full trace, Ⓢ pass count and breakpoint program testing. Has backtrace and histogram utilities. When used with MAC, provides full symbolic display of memory labels and equated values .**$105/$15**

☐ **ZSID**-Z80—Symbolic debugger with all fea- Ⓢ tures of SID ..... **$130/$15** ②

☐ **TEX**—Text output formatter to create paginat- Ⓢ ed, page-numbered and justified copy. Output can be directed to printer or disk ... **$105/$15**

☐ **DESPOOL**—Utility program to permit simulta- Ⓢ neous printing from text files while executing other programs ..... **$80/$10**

☐ **tiny C**—Interactive interpretive system for ⓧ teaching structured programming techniques. Manual includes full source listings .**$105/$50**

☐ **BDS C COMPILER**—Supports most features Ⓜ of language, including Structures, Arrays, ⓘ Pointers, recursive function evaluation, overlays. Includes linking loader, library manager, and library containing general purpose, file I/O, and floating point functions. Lacks initializers, statics, floats and longs. Documentation includes "The C PROGRAMMING LANGUAGE" by Kernighan and Ritchie ..... **$145/$25**

☐ **WHITESMITHS C COMPILER**—The ultimate Ⓛ in systems software tools. Produces faster ⓧ code than a pseudo-code Pascal with more extensive facilities. Conforms to the full UNIX* Version 7 C language, described by Kernighan and Ritchie, and makes available over 75 functions for performing I/O, string manipulation and storage allocation. Linkable to Microsoft REL files. Requires 60K CP/M ..... **$630/$30**

MICROSOFT

☐ **BASIC-80**—Disk Extended BASIC, ANSI Ⓛ compatible with long variable names, Ⓜ WHILE/WEND, chaining, variable length file records ..... **$325/$25**

☐ **BASIC COMPILER**—Language compatible Ⓒ with BASIC-80 and 3-10 times faster execution. Ⓜ Produces standard Microsoft relocatable binary output. Includes MACRO-80. Also linkable to FORTRAN-80 or COBOL-80 code modules ..... **$350/$25**

Software with Manual / Manual Alone

☐ **FORTRAN-80**—ANSI 66 (except for COM- Ⓛ PLEX) plus many extensions. Includes relocat- Ⓜ able object compiler, linking loader, library with manager. Also includes MACRO-80 (see below) ..... **$425/$25**

☐ **COBOL-80**—Level 1 ANSI '74 standard Ⓒ COBOL plus most of Level 2. Full sequential, Ⓜ relative, and indexed file support with variable file names. STRING, UNSTRING, COMPUTE, VARYING/UNTIL, EXTEND, CALL, COPY, SEARCH, 3-dimensional arrays, compound and abbreviated conditions, nested IF. Powerful interactive screen-handling extensions. Includes compatible assembler, linking loader, and relocatable library manager as described under MACRO-80 ..... **$700/$25**

☐ **MACRO-80**—8080/Z80 Macro Assembler. Ⓒ Intel and Zilog mnemonics supported. Relocat- Ⓜ able linkable output. Loader, Library Manager and Cross Reference List utilities included ..... **$149/$15**

☐ **muSIMP/muMATH**—muSIMP is a high level New Ⓛ programming language suitable for symbolic and semi-numerical processing. Implemented using a fast and efficient interpreter requiring only 7K bytes of machine code. muMATH is a package of programs written in muSIMP. The package performs sophisticated mathematical functions. Keeps track of up to 611 digits. Performs matrix operations on arrays: transpose, multiply, divide, inverse and other integer powers. Logarithmic, exponential, trigonometric simplification and transformation, symbolic differentiation with partial derivatives, symbolic integration of definite and indefinite integrals. Requires 40K CP/M. ..... **$250/$20**

☐ **muLISP-79**—Microcomputer implementation New Ⓛ of LISP. The interpreter resides in only 7K bytes of memory yet includes 83 LISP functions. Has infinite precision integer arithmetic expressed in any radix from 2 to 36. muLISP-79 includes complete trace facility and a library of useful functions and entertaining sample programs. ..... **$200/$15**

☐ **XMACRO-86**—8086 cross assembler. All Ⓛ Macro and utility features of MACRO-80 package. Mnemonics slightly modified from Intel ASM86. Compatibility data sheet available ..... **$275/$25**

☐ **EDIT-80**—Very fast random access text editor Ⓛ for text with or without line numbers. Global and intra-line commands supported. File compare utility included. ..... **$89/$15**

☐ **PASCAL/M***—Compiles enhanced Standard ⓧ Pascal to compressed efficient Pcode. Totally CP/M compatible. Random access files. Both 16 and 32-bit Integers. Runtime error recovery. Convenient STRINGs. OTHERWISE clause on CASE. Comprehensive manual (90 pp. indexed). SEGMENT provides overlay structure. INPORT, OUTPORT and untyped files for arbitrary I/O. Requires 56K CP/M. Specify 1) 8080 CP/M, 2) Z80 CP/M, or 3) Cromemco CDOS. ..... **$175/$20**

☐ **PASCAL/Z**—Z80 native code PASCAL com- ⓘ piler. Produces optimized, ROMable re-entrant ② code. All interfacing to CP/M is through the support library. The package includes compiler, relocating assembler and linker, and source for all library modules. Variant records, strings and direct I/O are supported. Requires 56K CP/M ..... **$395/$25**

☐ **PASCAL/MT**—Subset of standard PASCAL. Ⓜ Generates ROMable 8080 machine code. ⓧ Symbolic debugger included. Supports interrupt procedures, CP/M file I/O and assembly language interface. Real variables can be BCD, software floating point, or AMD 9511 hardware floating point. Includes strings enumerations and record data types. Manual explains BASIC to PASCAL conversion. Requires 32K ..**$250/ $30**

☐ **APL/V80**—Concise and powerful language for Ⓛ application software development. Complex ② programming problems are reduced to simple New expressions in APL. Features include up to 27K active workspace, shared variables, arrays of up to 8 dimensions, disk workspace and copy object library. The system also supports auxiliary processors for interfacing I/O ports. Requires 48K CP/M and serial APL printing terminal or CRT ..... **$500/$30**

☐ **ALGOL-60**—Powerful block-structured lan- ② guage compiler featuring economical run-time dynamic allocation of memory. Very compact (24K total RAM) system implementing almost all Algol 60 report features plus many powerful extensions including string handling direct disk address I/O etc. ..... **$199/$20**

☐ **CBASIC-2** Disk Extended BASIC—Non- Ⓜ interactive BASIC with pseudo-code compiler and run-time interpreter. Supports full file control, chaining, integer and extended precision variables, etc. ..... **$120/$15**

MICRO FOCUS

☐ **STANDARD CIS COBOL**—ANSI '74 COBOL Ⓛ standard compiler fully validated by U.S. Navy tests to ANSI level 1. Supports many features to level 2 including dynamic loading of COBOL modules and a full ISAM file facility. Also, program segmentation, interactive debug and powerful interactive extensions to support protected and unprotected CRT screen formatting from COBOL programs used with any dumb terminal ..... **$850/$50**

Software with Manual / Manual Alone

☐ **FORMS 2**—CRT screen editor. Output is Ⓛ COBOL data descriptions for copying into CIS COBOL programs. Automatically creates a query and update program of indexed files using CRT protected and unprotected screen formats. No programming experience needed. Output program directly compiled by STANDARD CIS COBOL ..... **$200/$20**

EIDOS SYSTEMS

☐ **KISS**—Keyed Index Sequential Search. Offers Ⓛ complete Multi-Keyed Index Sequential and Direct Access file management. Includes built-in utility functions for 16 or 32 bit arithmetic, string/integer conversion and string compare. Delivered as a relocatable linkable module in Microsoft format for use with FORTRAN-80 or COBOL-80, etc. ..... **$335/$23**

☐ **KBASIC**—Microsoft Disk Extended BASIC Ⓛ version 4.51 integrated by implementation of nine additional commands in language. Package includes KISS. REL as described above, and a sample mail list program. ....**$585/$45** To licensed users of Microsoft BASIC-80 (MBASIC) ..... **$435/$45**

☐ **XYBASIC** Interactive Process Control BASIC—Full disk BASIC features plus unique commands to handle byte rotate and shift and to test and set bits. Available in several versions:

Integer ROM squared	$350/$25
Integer CP/M	$350/$25
Extended ROM squared	$450/$25
Extended CP/M	$450/$25
Extended Disk CP/M	$550/$25
Integer CP/M Run Time Compiler	$350/$25
Extended CP/M Run Time Compiler	$450/$25

☐ **RECLAIM**—A utility to validate media under CP/M. Program tests a diskette or hard disk surface for errors, reserving the imperfections in invisible files, and permitting continued usage of the remainder. Essential for any hard disk. Requires CP/M version 2. ..... **$80/$5**

☐ **BASIC UTILITY DISK**—Consists of: (1) Ⓜ CRUNCH-14— Compacting utility to reduce the size and increase the speed of programs in Microsoft BASIC 4.51, BASIC-80 and TRS-80 BASIC. (2) DPFUN—Double precision subroutines for computing nineteen transcendental functions including square root, natural log, log base 10, sine, arc sine, hyperbolic sine, hyperbolic arc sine, etc. Furnished in source on diskette and documentation ..... **$50/$35**

☐ **STRING/80**—Character string handling plus routines for direct CP/M BDOS calls from FORTRAN and other compatible Microsoft languages. The utility library contains routines that enable programs to chain to a COM file, retrieve command line parameters and search file directories with full wild card facilities. Supplied as linkable modules in Microsoft format. **$95/$20**

☐ **STRING/80** source code available separately— **$295/NA**

☐ **THE STRING BIT**—FORTRAN character Ⓜ string handling. Routines to find, fill, pack, move, separate, concatenate and compare character strings. This package completely eliminates the problems associated with character string handling in FORTRAN. Supplied with source ..... **$65/$15**

☐ **VSORT**—Versatile sort/merge system for fixed Ⓜ length records with fixed or variable length fields. VSORT can be used as a stand-alone package or loaded and called as a subroutine from CBASIC-2. When used as a subroutine, VSORT maximizes the use of buffer space by saving the TPA on disk and restoring it on completion of sorting. Records may be up to 255 bytes long with a maximum of 5 fields. Upper/ lower case translation and numeric fields supported. ..... **$175/$20**

☐ **CPM/374X**—Has full range of functions to create or re-name an IBM 3741 volume, display directory information and edit the data set contents. Provides full file transfer facilities between 3741 volume data sets and CP/M files ..... **$195/$10**

Coming Soon

CPAids*

☐ **MASTER TAX**—Professional tax preparation Ⓛ program. Prepares schedules A, B, C, D, E, F, † G, R/RP, SE, TC, ES and forms 2106, 2119, 2210, 3468, 3903, 2441, 4625, 4726, 4797, 4972, 5695 and 6521. Printing can be on readily available, pre-printed continuous forms, on overlays, or on computer generated, IRS approved forms. Maintains client history files and is interactive with CPAids GENERAL LEDGER II (see below) ..... **$995/$30**

☐ **STANDARD TAX**—As above for schedules A, † B, C, D, E, G, R/RP, SE, TC and forms 2106 and 2441. Also, does not maintain client history files ..... **$495/$30**

☐ **GENERAL LEDGER II**—Designed for CPA's. † Stores complete 12 month detailed history of transactions. Generates financial statements, depreciation, loan amortizations, journals, trial balances, statements of changes in financial position, and compilation letters. Includes payroll system with automatic posting to general ledger. Prints payroll register, W2's and payroll checks. ..... **$450/$30**

Lifeboat Associates, 1651 Third Avenue, N.Y., N.Y. 10028 **(212) 860-0300** Telex: 220501
Neu in der Schweiz Lifeboat Associates GmbH, Aegeristr. 35, 6340 Baar **Telefon 042/31 2931**

Software with Manual / Manual Alone

☐ **T/MAKER**—Powerful new tool for preparing management reports with tabular data. Makes financial modeling projects easy. Do you want a weekly profitability report? Set up the table and compute. Just change the sales figures for next week and compute. You have a new report! T/MAKER includes a full screen editor for setting up tables which pages left, right, up and down. Compute includes standard arithmetic, percents, exponents, common transcendental functions, averages, maxima, minima, projections, etc. Requires 48K CP/M and CBASIC-2 ..... **$275/$25**

☐ **BSTAM**—Utility to link one computer to another Ⓜ also equipped with BSTAM. Allows file transfers at full data speed (no conversion to hex), with CRC block control check for very reliable error detection and automatic retry. We use it! It's great! Full wildcard expansion to send *.COM, etc. 9600 baud with wire. 300 baud with phone connection. **Both** ends need one. Standard and Ⓜ versions can talk to one another. **.$150/$10**

☐ **WHATSIT?***—Interactive data-base system using associative tags to retrieve information by subject. Hashing and random access used for fast response. Requires CBASIC-2 **.$175/$25**

☐ **SELECTOR III-C2**—Data Base Processor to † create and maintain multi-key data bases. Ⓜ Prints formatted sorted reports with numerical summaries or mailing labels. Comes with sample applications, including Sales Activity, Inventory, Payables, Receivables, Check Register, and Client/Patient Appointments, etc. Requires CBASIC-2. Supplied in source ..... **$295/$20**

☐ **GLECTOR**—General Ledger option to SELECTOR III-C2. Interactive system provides for customized COA. Unique chart of transaction types insure proper double entry bookkeeping. Generates balance sheets, P&L statements and journals. Two year record allows for statement of changes in financial position report. Supplied in source. Requires SELECTOR III-C2. CBASIC-2 and 56K system. ..... **$350/$25**

☐ **CBS**—Configurable Business System is a comprehensive set of programs for defining custom data files and application systems without using a programming language such as BASIC, FORTRAN, etc. Multiple key fields for each data file are supported. Set-up program customizes system to user's CRT and printer. Provides fast and easy interactive data entry and retrieval **with transaction processing**. Report generator program does complex calculations with stored and derived data, record selection with multiple criteria, and custom formats. Sample inventory and mailing list systems included. **No support language required** ..... **$295/$40**

MICRO DATA BASE SYSTEMS

☐ **HDBS**—Hierarchical Data Base System. CODASYL oriented with FILEs, SETs, RECORDs and ITEMs which are all user defined. ADD, DELETE, UPDATE, SEARCH, and TRAVERSE commands supported. SET ordering is sorted, FIFO, LIFO, next or prior. One to many set relationship supported. Read/write protection at the FILE level. Supports FILEs which extend over multiple floppy or hard disk devices.

☐ **MDBS**—Micro Data Base System. Full network data base with all features of HDBS plus multi-level read/write protection for FILE, SET, RECORD and ITEM. Explicit representation of one to one, one to many, many to many, and many to one SET relationships. Supports multiple owner and multiple record types within SETs. HDBS files are fully compatible.

☐ **HDBS-Z80** version ..... **$250/$40****

☐ **MDBS-Z80** version ..... **$750/$40****
8080 version available at $75 extra.

When ordering, specify one of the language interfaces listed below. Additional language interfaces available at time of purchase for $100 or $125 if purchased later.

**The single manual covering HDBS and MDBS when purchased alone comes without specific language interface manual. Manuals are available for the following Microsoft languages:

1) MBASIC 4.51, 2) BASIC-80 5.0, 3) Compiled BASIC or FORTRAN-80, 4) COBOL-80, 5) MACRO-80. ..... **$NA/$10**

MICROPRO

☐ **SUPER-SORT I**—Sort, merge, extract utility as Ⓛ absolute executable program or linkable module in Microsoft format. Sorts fixed or variable records with data in binary, BCD, Packed Decimal, EBCDIC, ASCII, floating & fixed point, exponential, field justified, etc. Even variable number of fields per record! ..... **$225/$25**

☐ **SUPER-SORT II**—Above available as abso-Ⓛ lute program only ..... **$175/$25**

☐ **SUPER-SORT III**—As II without SELECT/ Ⓛ EXCLUDE ..... **$125/$25**

☐ **DATASTAR**—Professional forms control entry Ⓛ and display system for key-to-disk data capture. Menu driven with built-in learning aids. Input field verification by length, mask, attribute (i.e. upper case, lower case, numeric, auto-dup, etc.). Built-in arithmetic capabilities using keyed data, constant and derived values. Visual feedback for ease of forms design. Files compatible with CP/M-MP/M supported languages. Requires 32K CP/M ..... **$350/$35**

Software with Manual / Manual Alone

☐ **WORD-STAR**—Menu driven visual word pro-Ⓛ cessing system for use with standard terminals. Text formatting performed on screen. Facilities for text paginate, page number, justify, center and underscore. User can print one document while simultaneously editing a second. Edit facilities include global search and replace, Read/Write to other text files, block move, etc. Requires CRT terminal with addressable cursor positioning ..... **$445/$40**

☐ **WORD-STAR-MAIL-MERGE**—As above with Ⓛ option for production mailing of personalized documents with mail lists from DATASTAR or NAD ..... **$575/$40**

☐ **WORD-STAR Customization Notes**—For sophisticated users who do not have one of the many standard terminal or printer configurations in the distribution version of WORDSTAR ..... **$NA/$95**

☐ **WORD-MASTER** Text Editor—In one mode Ⓛ has superset of CP/M's ED commands including global searching and replacing, forwards and backwards in file in video mode, provides full screen editor for users with serial addressable-cursor terminal ..... **$145/$25**

☐ **TEXTWRITER III**—Text formatter to justify and Ⓜ paginate letters and other documents. Special features include insertion of text during execution from other disk files or console, permitting recipe documents to be created from linked fragments on other files. Has facilities for sorted index, table of contents and footnote insertions. Ideal for contracts, manuals, etc. Now compatible with Electric Pencil* and Word-Star prepared files ..... **$125/$20**

New lower prices for application Software

PEACHTREE SOFTWARE

☐ General accounting software for small busi-Ⓛ nesses. Each product can be used alone or with † automatic posting to the general ledger. Supplied in source for Microsoft BASIC 4.51.

GENERAL LEDGER	**$530/$40**
ACCOUNTS PAYABLE	**$530/$40**
ACCOUNTS RECEIVABLE	**$530/$40**
PAYROLL	**$530/$40**
INVENTORY	**$660/$40**
ALSO:	
MAILING ADDRESS	**$530/$40**
PROPERTY MANAGEMENT	**$925/$40**

GRAHAM-DORIAN SOFTWARE SYSTEMS

☐ Comprehensive accounting software written in Ⓛ CBASIC-2 and supplied in source code. Each Ⓜ software package can be used as a stand-alone † system or integrated with the General Ledger for automatic posting to ledger accounts. Requires CBASIC-2.

GENERAL LEDGER	**$805/$40**
ACCOUNTS PAYABLE	**$805/$40**
ACCOUNTS RECEIVABLE	**$805/$40**
INVENTORY SYSTEM	**$555/$40**
JOB COSTING	**$805/$40**
APARTMENT MANAGEMENT	**$805/$40**
CASH REGISTER	**$805/$40**

☐ **POSTMASTER**—A comprehensive package Ⓜ for mail list maintenance that is completely menu driven. Features include keyed record extraction and label production. A form letter program is included which provides neat letters on single sheet or continuous forms. Includes NAD file translator. Requires CBASIC-2 ..... **$150/$20**

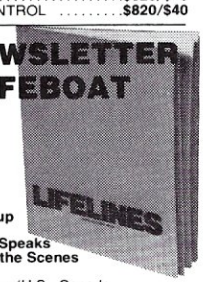
New Features *New Manual*

STRUCTURED SYSTEMS GROUP

☐ Complete interactive accounting software for † business. Each product can be used standalone or with automatic posting to the general ledger. Each product is thoroughly tested and very well documented. Each product requires CBASIC-2.

GENERAL LEDGER	**$820/$40**
ACCOUNTS RECEIVABLE	**$820/$40**
ACCOUNTS PAYABLE	**$820/$40**
PAYROLL	**$820/$40**
INVENTORY CONTROL	**$820/$40**

NEW! NEWSLETTER FROM LIFEBOAT



- **Latest Version Numbers List of Software**
- **Update on CP/M Users Group**
- **The Great ZOSO Speaks Out from Behind the Scenes**

$18 ppd. for 12 issues (U.S., Canada, Mexico). Elsewhere **$40.**
Send Check to "Lifelines," 1651 Third Avenue, New York, N.Y. 10028 or use your *VISA* or *Mastercharge*—call (212) 722-1700

Software with Manual / Manual Alone

☐ **ANALYST**—Customized data entry and report-† ing system. User specifies up to 75 data items per record. Interactive data entry, retrieval, and update facility makes information management easy. Sophisticated report generator provides customized reports using selected records with multiple level breakpoints for summarization. Requires a disk sort utility such as QSORT, SUPER-SORT or VSORT and CBASIC-2 ..... **$250/$15**

☐ **LETTERIGHT**—Program to create, edit and type letters or other documents. Has facilities to enter, display, delete and move text, with good video screen presentation. Designed to integrate with NAD for form letter mailings. Requires CBASIC-2 ..... **$200/$25**

☐ **NAD** Name and Address selection system—Interactive mail list creation and maintenance program with output as full reports with reference data or restricted information for mail labels. Transfer system for extraction and transfer of selected records to create new files. Requires CBASIC-2 ..... **$100/$20**

☐ **QSORT**—Fast sort/merge program for files with fixed record length, variable field length information. Up to five ascending or descending keys. Full back-up of input files created ..... **$100/$20**

★ ★ ★ ★ ★ ★ ★

CONDIMENTS

★ ★ ★ ★ ★ ★ ★

☐ **HEAD CLEANING DISKETTE**—Cleans the drive Read/Write head in 30 seconds. Diskette absorbs loose oxide particles, fingerprints, and other foreign particles that might hinder the performance of the drive head. Lasts at least 3 months with daily use. Specify 5" or 8".
Single sided ..... **$20 each/$55 for 3**
Double sided ..... **$25 each/$65 for 3**

☐ **FLIPPY DISK KIT**—Template and instructions to modify single sided 5¼" diskettes for use of second side in single sided drives **$12.50**

☐ **FLOPPY SAVER**—Protection for center holes for 5" and 8" floppy disks. Only 1 needed per diskette. Kit contains centering post, pressure tool and tough 7 mil mylar reinforcing rings for 25 diskettes.
5" Kit **$14.95**
5" Rings only **$7.95**
8" Kit **$16.95**
8" Rings only **$8.95**

☐ **PASCAL USER MANUAL AND REPORT**—By Jensen and Wirth. The standard textbook on the language. Recommended for use by Pascal/Z, Pascal/M and Pascal/MT users **$12**

☐ **THE C PROGRAMMING LANGUAGE**—By Kernighan and Ritchie. The standard textbook on the language. Recommended for use by BDS C, tiny C, and Whitesmiths C users . . **$12**

☐ **STRUCTURED MICROPROCESSOR PROGRAMMING**—By the authors of SMAL/80. Covers structured programming, the 8080/8085 instruction set and the SMAL/80 language **$20**

☐ **ACCOUNTS PAYABLE & ACCOUNTS RECEIVABLE—CBASIC**—By Osborne/McGraw-Hill **$20**

☐ **GENERAL LEDGER—CBASIC**—By Osborne/McGraw-Hill **$20**

☐ **PAYROLL WITH COST ACCOUNTING—CBASIC**—by Osborne/McGraw-Hill **$20**

☐ **LIFEBOAT DISK COPYING SERVICE**—Transfer data or programs from one media format to another at a moderate cost . . **from $25**

★ ★ ★ ★ ★ ★ ★

Hearty Appetite.

★ ★ ★ ★ ★ ★ ★

*CP/M and MP/M are trademarks of Digital Research.
Z80 is a trademark of Zilog, Inc.
UNIX is a trademark of Bell Laboratories.
WHATSIT? is a trademark of Computer Headware.
Electric Pencil is a trademark of Michael Shrayer Software.
TRS-80 is a trademark of Tandy Corp.
Pascal/M is a trademark of Sorcim.
SoftCard is a trademark of Microsoft.
Apple is a trademark of Apple Computer.
PASM, PLINK, BUG and µBUG are trademarks of Phoenix Software Associates Ltd.
CPAids is a trademark of Computer Tax Service, Inc.

† Recommended system configuration consists of 48K CP/M, 2 full size disk drives, 24 x 80 CRT and 132 column printer.

Ⓜ Modified version available for use with CP/M as implemented on Heath and TRS-80 Model I computers.

Ⓛ User license agreement for this product must be signed and returned to Lifeboat Associates before shipment may be made.

Ⓘ This product Includes/eXcludes the language Ⓧ manual recommended in Condiments.

Ⓢ Serial number of CP/M system must be supplied with orders.

Ⓩ Requires Z80 CPU.

Ordering Information

MEDIA FORMAT ORDERING CODES
When ordering, please specify format code.

LIFEBOAT ASSOCIATES MEDIA FORMATS LIST
Diskette, cartridge disk and cartridge tape format codes to be specified when ordering software for listed computer or disk systems. All software products have specific requirements in terms of hardware or software support, such as MPU type, memory size, support operating system or language.

Computer system	Format Code
Altair 8800 Disk	See MITS 3200
Altos	A1*
Apple + SoftCard 13 Sector	RG
Apple + SoftCard 16 Sector	RR
BASF System 7100	RD
Blackhawk Single Density	Q3
Blackhawk Micropolis Mod II	Q2
CDS Versatile 3B	Q1
CDS Versatile 4	Q2
COMPAL-80	Q2
Cromemco System 3	A1*
Cromemco Z2D	R6
CSSN BACKUP (tape)	T1 #
Delta	A1*
Digi-Log Microterm II	RD
Digital Microsystems	A1*
Discus	See Morrow Discus
Durango F-85	RL
Dynabyte DB8/2	R1
Dynabyte DB8/4	A1*
Exidy Sorcerer + Lifeboat CP/M	Q2
Exidy Sorcerer + Exidy CP/M	Q4
Heath H8 + H17/H27	P4
Heath H89 + Lifeboat CP/M	P4
Heath H89 + Magnolia CP/M	P7
Helios II	See Processor Technology
Horizon	See North Star
iCOM 2411 Micro Floppy	R3
iCOM 3712	A1
iCOM 3812	A1*
iCOM 4511 5440 Cartridge CP/M 1.4	D1 #
iCOM 4511 5440 Cartridge CP/M 2.2	D2 #
IMS 5000	RA
IMS 8000	A1*
IMSAI VDP-40	R4**
IMSAI VDP-42	R4**
IMSAI VDP-44	R5**
IMSAI VDP-80	A1**
Intecolor	See ISC Intecolor
Intel MDS Single Density	A1
Intertec SuperBrain DOS 0.1	R7
Intertec SuperBrain DOS 0.5-2.X	RJ
Intertec SuperBrain DOS 3.X	RK
ISC Intecolor 8063/8360/8963	A1
Kontron PSI-80	RF
Meca 5¼"	P6
Micromation (Except TRS-80 below)	A1*
Micropolis Mod I	Q1
Micropolis Mod II	Q2
MITS 3200/3202	B1
Morrow Discus	A1*
Mostek	A1
MSD 5¼"	RC
North Star Single Density	P1
North Star Double/Quad	P2
Nylac Single Density	Q3
Nylac Micropolis Mod. II	Q2
Ohio Scientific C3	A3
Onyx C8001	T2 #
Pertec PCC 2000	A1*
Processor Technology Helios II	B2
Quay 500	RO
Quay 520	RP
RAIR Single Density	R9
RAIR Double Density	RE
Research Machines 8"	A1
Research Machines 5¼"	RH
REX	Q3
Sanco 7000 5¼"	RQ
SD Systems 8"	A1*
SD Systems 5¼"	R3
Sorcerer	See Exidy Sorcerer
Spacebyte	A1
SuperBrain	See Intertec
Tarbell	A1*
TEI 5¼"	R3
TEI 8"	A1*
Thinkertoys	See Morrow Discus
TRS-80 Model I 5¼"	R2
TRS-80 Model I + FEC Freedom	RN
TRS-80 Model I + Micromation	A4*
TRS-80 Model I + Omikron 5¼"	RM
TRS-80 Model I + Omikron 8"	A1
TRS-80 Model I + Shuffleboard 8"	A1
TRS-80 Model II	A1*
VDP-40/42/44/80	See IMSAI
Vector MZ	Q2
Versatile	See CDS Versatile
Vista V80 5¼" Single Density	P5
Vista V200 5¼" Double Density	P6
Zenith Z89 + Lifeboat CP/M	P4
Zenith Z89 + Magnolia CP/M	P7

Prices F.O.B. New York. Shipping, handling and C.O.D. charges extra.

Manual cost applicable against price of subsequent software purchase.

The sale of each proprietary software package conveys a license for use on one system only.

VISA master charge

* Single-Side Single-Density disks are supplied for use with Double-Density and Double-Side 8" soft sector format systems.

** IMSAI formats are single density with directory offset of zero.

A media surcharge of $25 for orders on tape formats T1 and T2 and of $100 for orders on disk formats D1 and D2 will be added.

The list of available formats is subject to change without notice. In case of uncertainty, call to confirm the format code for any particular equipment.

™ The Software Supermarket is a trademark of Lifeboat Associates

Introduction to CP/M—Part IV

by Jake Epstein

In the previous articles in this series on the CP/M operating system, the focus has been on general concepts. With the first part of this article, I will conclude discussing introductory material and begin exploring more technical matters. Since I will be dealing with assembly language, those readers who have not had experience in this area might find it helpul to refer to one or more of the references listed at the end of this article.

*Reg Tmk Digital Research

CP/M Utilities

Digital Research supplies several utilities with CP/M. These various programs are used for file management, text processing, program development, and information transferral. Rather than discuss each program in depth, I will merely list them and give a brief description of their function. Certain utilities will be covered in depth in following articles when system alteration and program development are discussed. If anyone is interested in submitting articles devoted exclusively to the description and use of specific utilities and/or other CP/M programs, feel free to contact S-100 Microsystems.

ED.COM - This is a line oriented text editor used for preparation of ASCII files. With CP/M, it is usually used in co-ordination with ASM.COM and SUBMIT.COM, but can be also used to prepare text for other programs such as BASIC or a TEXT PROCESSOR.

ASM.COM - This program is used to assemble Intel 8080 standard format ASCII files. Input files have the secondary name ASM while output files have secondary names of HEX and/or PRN. HEX files, which are in Intel hex format, contain a representation of the machine code of the assembled source program. PRN files are identical to the ASM source file but with machine code equivalents listed in hexadecimal code adjacent to each line of assembly code.

LOAD.COM - Hex format files are converted to executable binary files using this program. The output file produced is a COM file which can then be executed using the CCP LOAD and EXECUTE command.

STAT.COM - File characteristics such as length or write protection are checked with STAT. Logical devices can also be checked using this program. The STAT utility is much more comprehensive in CP/M 2.0 and later versions than in earlier versions.

DDT.COM - This program provides an extensive array of commands that are used to inspect and modify the contents of system memory. It can be used to create, analyze, and debug programs with the following functions: 1. alter memory using hexadecimal numbers or Intel standard assembler mnemonics; 2. disassemble machine code to Intel Standard mnemonics; 3. dump areas of memory showing both hexadecimal equivalents and ASCII equivalents of each location; 4. execute a program with breakpoints; 5. list and alter the contents of processor registers and flags; 6. perform hexadecimal addition and subtraction; 7. perform single or mutiple step execution of programs showing machine register contents and flag status after each instruction; 8. load files from disk to any location in memory.

SUBMIT.COM - This program is used for batch processing. Using ED.COM, the user prepares a file that contains a list of CCP commands to be executed by the operating system. This file has the secondary name SUB. When SUBMIT is invoked, all the commands listed in the SUB file will be executed just as if they had been entered at the console. SUBMIT will only work when drive 0 (system drive) is logged in. In CP/M 1.4 and earlier versions, once the user leaves the CCP mode, as when executing a program, the SUBMIT function relinquishes control of the system until a warm boot (cntr-C) is executed by the user. In version 2.0 and later, commands that are interpreted by a program can be programmed using the XSUB.COM utility in co-ordination with SUBMIT. Thus, long and complex processes involving many different devices and programs can be executed with or without user interaction.

MOVCPM.COM - System alteration for various sized memories is accomplished using this utility. MOVCPM will be discussed in depth under system alteration in a future article in this series.

PIP.COM - The peripheral Interchange Program is used to transfer information from one location to another. The locations involved can be disk files or I/O devices such as the console or the printer. PIP also contains several software switches that allow for verification or alteration of the data flow.

SYSGEN.COM - This program is used to read or write to the system tracks of a diskette. This allows the operator to alter components of the operating system. This utility will also be discussed in depth under system alteration in a future article.

In addition to the COM files listed above, Digital Research provides two ASCII files that are used in alteration of a 2.0 or later version system. These files, DISKDEF.LIB and DEBLOCK.ASM, will be dicussed in

Jake Epstein, Rainbow Associate, 25 Sagamore Cove, Branford, CT 06405

the next article under BIOS II. The term TRANSIENT COMMAND is often used for utilities such as PIP and STAT because they are often used in a fashion similar to CCP commands to monitor and alter system status. Rather than being built into the CCP, however, they are loaded and then executed in the TPA, Transient Program Area.

BIOS - Part I

As discussed in the first article of this series, BIOS (Basic Input/Output System) is the the module of CP/M in which software interfaces to computer peripherals are located. The BIOS that is provided by Digital Research is for the Intel MDS INTELLEC computer system. If the user does not have this computer system, then the BIOS has to be modified for his/her hardware configuration. Once implemented however, software that is CP/M- compatible may be executed without need for additional modification of the operating system. Manufacturers of disk controller boards for 8080/Z80/8085-based microcomputers supply BIOS's, but the user may still have to modify non-disk routines such as serial or parallel port drivers. The main focus of this section and BIOS - Part II will be on the structure and operation of BIOS with a few examples of modification. This section will be devoted to general organization of BIOS and non-disk I/O routines. Part II will deal with disk I/O routines with emphasis on version 2.0 enhancements. It is advisable that the reader obtain a listing of a working BIOS to use as a reference to this section. Issue number 2 of S-100 MICROSYSTEMS contains an excellent BIOS for version 1.4 written by Martin Nichols (see references).

Structure

BIOS is the last module in the CP/M memory map. All versions of CP/M have the following routines:

1: COLD BOOT	This routine is used to initialize various areas of the operating system after the CP/M is loaded from the system diskette.
2: WARM BOOT	Used after a system reset (cntr-C) to load in CCP and BDOS without the need to load and initialize the entire system.
3: CONSOLE STATUS	Used to determine whether or not data is ready to be input from the console device.
4: CONSOLE INPUT	Used to input data from the console.
5: CONSOLE OUTPUT	Used to transmit data to the console device.
6: LIST OUTPUT	Used to transmit data to the list (hard- copy) device.
7: PUNCH	Used to transmit data to the paper tape punch.
8: READER	Used to input data from the paper tape reader.
9: HOME	Causes the logged in disk drive to seek track 00.
10: SELECT DISK	Used to select the disk drive where disk I/O will take place. Used to log in a disk.
11: SET TRACK	Initializes the disk controller to a specified track.
12: SET SECTOR	Initializes the disk controller to a specified sector.
13: SET DMA ADDRESS	Sets the beginning of a buffer area in system memory where data transferral will occur during disk I/O.
14: READ	Data will be transferred from the disk to the DMA buffer as determined by the above routines. The amount of data is that which is contained in one physical unit on the disk - one sector of 128 bytes on a single density floppy diskette.
14: WRITE	Same as read only data is transferred from the DMA buffer to the disk.

The following two routines have been added to CP/M 2.0 and later versions:

15: LIST STATUS	Used to check status of the list device.
16: SECTOR	Used to convert from a logical location TRANSLATE to a physical location of a sector. Used with translation tables found elsewhere in BIOS. See Article II in this series for a preliminary discussion of sector skew.

At the beginning of BIOS, a vector jump table is located to direct programs to the various routines. The various jumps have to be placed in the order given above, but the routines themselves may be anywhere. There are three different ways in which user programs may interface to I/O devices. 1. The actual software drivers may be part of the program. In this case the program is not useable in systems different from the development system without modification. 2. The program may use system calls to BDOS to accomplish I/O. In this case, BDOS and BIOS transfer data as determined by one of 36 numbers placed in general microprocessor register 'C'. 3. A final possibility is in direct calls to the appropriate routine in BIOS. This is useful when a desired I/O function is not implemented in BDOS. An example of this is console input. In CP/M 1.4 and earlier versions, any input via a console input system call will cause the data to be transmitted back to the screen. In order to eliminate this character echo, one merely needs to call the fourth jump vector, console input, of the BIOS jump table. One disadvantage to this procedure is that the calls from the user program must be done in a manner that is compatible with different sized CP/M systems. See

Intro to CP/M cont'd...

listing 1 for an example of how this can be accomplished. A final note; additional routines may be added to BIOS, but jump vectors must be placed at the end of the jump table so that its continuity is not upset.

Besides the above routines, various implementations may have areas for temporary storage or data buffering. In CP/M 2.0 and later versions, areas of BIOS are reserved for disk information. Each disk unit in the system has a disk parameter block and a sector translation table stored in BIOS. This gives the operating system the ability to handle diferent types of mass storage units. The disk parameter block contains information such as disk size, sector size, sectors per track, etc. The sector translation table is used to determine sector skew. These areas are used by BDOS to calculate physical sector locations from logical file information. Also, BIOS contains a buffer area for directory operations. These topics will be discussed at length in BIOS - PART II.

At this point I will go into more depth on the CP/M memory map. This information is important in understanding some of the parameters influencing the location and size of the BIOS. It will also serve to introduce material that will be expanded in the article dealing with system alteration. The following table shows the size and relative locations in a minimal-size CP/M memory map.

MODULE	SECTORS	STARTING ADDRESS	SIZE IN BYTES		
{Map of 16K CP/M 1.4 System}					
---	--	00H	100H	=	256
TPA	--	100H	2800H	=	10240
CCP	16	2900H	800H	=	2048
BDOS	26	3100H	0D00H	=	3328
BIOS	4	3E00H	200H	=	512
Totals	46		4000H	=	16384
{Map of 20K CP/M 2.0 System}					
---	--	00H	100H	=	56
TPA	--	100H	3300H	=	13056
CCP	16	3400H	800H	=	2048
BDOS	28	3C00H	E00H	=	3584
BIOS	7	4A00H	380H	=	896
Totals	51		4C80H	=	19840

The minimal memory size configuration available in CP/M is 16K in 1.4 and earlier versions and 20K in CP/M 2.0 and later versions. For larger size-configurations, the CCP and BDOS have to be relocated using MOVCPM, while BIOS and the COLD START LOADER have to modified and reassembled to the new system memory size. Although the CCP, BDOS, and BIOS remain in the same relative locations, the TPA is either expanded or contracted to fill out added or deleted space in various sized systems. In the tables, the TPA begins at 100H and is either 2800H or 3300H long depending on version. Adding 100H to either of these values gives the staring point of the CCP: 2900H in 1.4 and 3400H in 2.0. To calculate where the CCP would start in other sized systems, simply use the following formula:

FCCP Start (CBASE) = TPA size + (BIAS * 400H)
where: BIAS = Memory size - Minimal configuration {in K bytes}
400H = 1024 or 1K bytes

BIAS is actually the number of 1K segments that the memory is greater than the minimal configuration. The following equations should clarify this material.

For a 24K 1.4 System:

BIAS = 24K-16K=8K
CCP START = 2900H + (8*400H) = 2900H + 2000H = 4900H

For a 62K 2.0 System:

BIAS = 62K - 20K = 42K
CCP START = 3400H + (42*400H) = 3400H + A800H = DC00H

To calculate the location of BIOS simply add the combined size of BDOS + CCP to the CCP starting location. The following tables give the CCP and BIOS starting points for various systems.

16K	0	2900H	3E00H				
17K	1	2D00H	4200H				
18K	2	3100H	4600H				
19K	3	3500H	4A00H				
20K	4	3900H	4E00H	:	0	3400H	4A00H
21K	5	3D00H	5200H	:	1	3800H	4E00H
22K	6	4100H	5600H	:	2	3A00H	5200H
23K	7	4500H	5A00H	:	3	3E00H	5800H
24K	8	4900H	5E00H	:	4	4400H	5A00H
28K	12	5900H	6E00H	:	8	5400H	6A00H
32K	16	6900H	7E00H	:	12	6400H	7A00H
40K	24	8900H	9E00H	:	20	8400H	9A00H
48K	32	A900H	BE00H	:	28	A400H	BA00H
56K	40	C900H	DE00H	:	36	C400H	DA00H
64K	48	E900H	FE00H	:	44	E400H	FA00H

A problem can arise when dealing with different implementations of BIOS. Notice that in a 1.4 system there are only 512 bytes of memory, xE00H-xFFFH, available for BIOS. Since most BIOS modules will be larger in size, especially when long drivers such as video routines are added, certain functions will have to be placed in ROM and then called from BIOS. Since the system diskette has a total of 9 sectors available -- the total number on tracks 0 and 1 minus the total needed by the COLD START Loader, CCP , and BDOS -- BIOS can be a total of 9 * 128 or 1152 bytes long. To build a working system, BIOS could start at location xA00H. To do this, the system would be built one K smaller than avaialble memory. For example, for a system with 24K of available memory, a 23k CP/M system would be built using the MOVCPM utility, and BIOS and the COLD START LOADER would be reassembled for the proper CCP, BDOS, and BIOS starting points. To complete the modification, the

number of sectors read by the COLD START LOADER and the WARM BOOT routine in BIOS would have to be adjusted. Often , distributors of CP/M systems may alter MOVCPM so that this 1K offset is handled automatically. Problems arise when a BIOS that has been built for the standard MOVCPM is added to one of these systems. The easiest solution to the problem is to simply build system a 1K smaller, and then proceed with the modification of CP/M in the normal manner.

In CP/M 2.0, the limitation is not in system memory size but in disk space. There are only 7 sectors (380H bytes) available on a single-density 8-inch disk for BIOS, and although this is enough storage for a minimal BIOS, any additions will quickly overflow this space. One way to get around the problem is to place all buffer and scratch RAM areas at the end of BIOS. Thus only permanent code need be loaded. If certain RAM locations need to be initialized, then routines will have to be placed in the COLD BOOT module of BIOS to accomplish this. This solution could also make the permanent code larger than 7 sectors. Another option is the placing of certain routines in ROM. The easiest and most common method would be to place the ROM in the last area of memory, usually F000H or F800H. Since most routines will need stack space and scratch areas, it is wise to have RAM located in this area also. Some of the newer ROM boards have space for RAM, but if this option isn't available, then space will have to be reserved in an area outside the bounds of CP/M in system memory. A problem with this configuration is that any time system memory is added and/or CP/M is enlarged, the ROM may have to be erased and reburned if it is an EPROM; or a brand new ROM purchased if it is not. There are ways of computing locations using elaborate routines in software, but the easiest way is the one using a ROM/RAM board. Note: the Z80 and the new 16 bit microprocessors (Z8000, 8086, 68000) do provide relative and indexed addressing schemes that would help circumvent this problem.

Another solution is in configuring the system so that there are more than 2 system tracks. This can be done easily in CP/M 2.0 by using the DISKDEF.LIB file and the CP/M MACRO Assembler. I will discuss this in BIOS-PART II. A problem with this is that software supplied from other sources may not be compatible with the modified system. To solve this problem, having a logical device that can read standard diskettes could be implemented in BIOS. As an example, in a two-drive system, devices A: and B: could be interfaced to drives 0 and 1 with their configuration being the one most commonly used in the system. Logical device C:, on the other hand, could be set to access drive 1 with its configuration being different than A: and B:. Using PIP, software could then be transferred between A: and C:.

A final method would be to have a minimal BIOS stored on the system tracks with an expanded BIOS stored as a file. Upon COLD START, this file could be ovelaid in memory either manually or automatically. CP/M 1.4 and 2.0 can be modified so that a file is loaded and executed upon COLD BOOT. I will discuss this option in the article on system calls.

BIOS - Modules

The following discussion decribes the parameters of non-disk I/O modules in depth.

CONSOLE STATUS: Upon return from this routine, the value 00 in register 'A' indicates that no character has been input at the console device; 0FFH indicates a character has been input.

CONSOLE INPUT: Upon return, register 'A' will contain an input character. The most significant bit (parity bit) should be reset to zero. This is accomplished most easily with the operation ANI 7FH. Routines may be added to convert characters to different values. For example: delete (7FH) backspace (08H).

CONSOLE OUTPUT: Upon entry, register 'C' contains the character to be printed. The 'A' register will be changed except where Z80 instructions are used. Routines may be placed to control output.

READER: Same parameters as CONSOLE INPUT. May be modified to function with other devices such as modems for telephone communication.

PUNCH: Same parameters as CONSOLE OUTPUT.

LIST: Same parameters as CONSOLE OUTPUT. Routines can be added to control printer paging to or route (SPOOL) data to a disk file.

LIST STATUS: Same as CONSOLE STATUS. Used with SPOOL or DESPOOL programs.

Programmed I/O

Martin Nichols' BIOS serves as an excellent example of how to implement the above routines. In all cases, there are four possible ways of programming these routines. The most common and easiest approach is polled I/O. In this method, I/O is accomplished only when the routines are called. In order for each routine to operate properly, status has to be checked for each device, with program loops being used to force the computer to wait until the device is ready for I/O. In other words, the routine polls the I/O device's status register to determine when it is ready to send or receive data. In most commercially prepared BIOS's, options that can be selected at assembly time are given for various boards. If a board is not covered, then then the user must program into the appropriate routine the location and the logic of the status register and the location of the DATA register. A BIOS routine that allows no I/O usually indicates an error in port location. A quick stream of characters with only one input or output usually implies improper status logic or bit location. The following routine shows how to program an I/O port:

```
CONOT:                  ;CONSOLE OUT ROUTINE
        IN    00        ;INPUT STATUS FROM PORT 0
        ANI   02        ;CHECK BIT BY ANDING 'A' WITH 0000 0001B
        JZ    CONOT     ;IF 'A' REGISTER WAS 0000 0000B THEN LOOP
                        ;IF NOT GO ON (JZ MEANS JUMP IF ZERO)
        MOV   A,C       ;PUT CHARACTER IN 'A' REGISTER
        OUT   01        ;OUTPUT CHARACTER IN 'A' TO PORT 1
        RET             ;RETURN TO THE CALLING PROGRAM
```

In this routine, the logic is positive because the status bit must go from 0 to 1 to indicate that the device is ready. Negative logic is that case where the bit changes in a negative direction (1 to 0). In a negative logic system, the third instruction would be JNZ (JUMP if not zero).

Intro to CP/M cont'd...

A second method is memory mapped-I/O. This technique is almost identical to polled I/O except that memory access instructions are used instead of inputs and outputs. Although programming using this method can be quite efficient, a major disadvantage is that I/O ports use system memory space. An example of a memory mapped I/O scheme is the keyboard of the Radio Shack TRS-80 microcomputer.

Interrupts

Another method is interrupt-driven I/O. In this case, when a device is ready to perform I/O, it interrupts the computer from its current task so that it (the computer) can perform the operation. This eliminates the need for the computer to sit and wait while it checks status via a loop as in the above example. Also, interrupts can be used to control program execution. Suppose that you have written a program that outputs integers to the screen. If it is written as an infinite loop, then the only way to halt execution may be a system reset. If, as a part of an interrupt-driven I/O routine, input is checked for a certain character such as ESCape (1BH), program execution could be broken by pressing the appropriate key. One of the main uses of the CONSOLE STATUS routine is for the same type of function.

The main problem with interrupts are that they are much harder to implement than polled I/O. This is true for both hardware and software. To understand what is involved, I will devote a bit of discussion to the hardware and software interfacing of the Intel 8080 microproccesor. I suggest consulting the Intel 8080 Users manual and the articles on the proposed IEEE S-100 standard that are listed under the reference section at the end of this article for diagrams, timing charts, and descriptions.

Before an interrupt can occur, the 8080 must be "software- initialized" (instruction is part of a program) using the ENABLE INTERRUPT (EI - FBH) instruction. Upon initialization, a flag inside the 8080 is set to indicate that interrupts are enabled. To interrupt the computer, the I/O interface logic pulls pin 14 high on the 8080. This pin is connected to line 73 of the S-100 bus, but it is interfaced so that the I/O device has to pull it low to cause an interrupt. After this occurs, the computer will finish its current instruction and then service the interrupt. The interrupting hardware provides the next machine instruction instead of program memory as pointed to by the 'PC' (program counter) which normally is the case. This is accomplished via an 8080 status signal that indicates an interrupt read instead of a memory read. This signal is supplied by S-100 line 96, as opposed to line 47, which indicates a memory read.

Although any instruction may be supplied, the usual procedure is to use one of the 8 RST (restart) instructions. These instructions have two effects. First, they all cause the value of the 'PC' to be saved on the processor stack with the usual update of register 'SP' (the stack pointer), as is the case with a CALL instruction. Then, depending on the instruction, the program will jump to one of 8 locations: 0, 08H, 10H, 18H, 20H, 28H, 30H, 38H. With CP/M, three of these locations cannot be used. Location 00, which is restarted by instuction RST 00 - C7H, is reserved for the warm boot jump vector, as discussed in Article 3 of this series. Location 30H, RST 6 - F7H, is the first location of an 8-byte block reserved for future use by Digital Research. Finally, location 38H, RST 7- FFH, is used by utility programs such as DDT to control and/or monitor program execution. By placing a RST 7 instruction in place of another instruction and saving the replaced instruction and its location, a program can be run using the computer itself to execute the program up to the instruction. At that time the computer breaks out of the program and control returns to the utility, which may or may not return the initially changed instructions to the program. In DDT, two RST 7 instructions may be placed in the program at one time and are called "breakpoints".

Although the RST instruction may be supplied by the I/O port interface, often a separate interface board will be used to supply the instruction. The I/O port will request an interrupt via one of 8 VECTOR INTERRUPT pins on the S-100 bus, pins 4-11. When activated, the vector interrupt interface will generate an interrupt through S-100 line 73 and then place the RST instruction on the data bus at the appropriate time. The CPU will indicate when it is ready for this transfer via line 96; this line is used in place of line 47, which indicates a memory read. Each vector interrupt pin corresponds to a specific RST instruction with pin 4 referring to RST 0, pin 5 - RST 1 and so on. The vector interrupt interface may also contain a priority encoder. This device allows the user to set up different priorities for vector interrupts. While an interrupt is being serviced, other requests may occur in systems with more than one interrupt-driven device. When the time comes, the device with the highest priority will be serviced next. In a system with more than one device, this facility is necesary to prevent timing errors. In my next article, I will include a diagram of a vector interrupt interface that I built to use interrupts with my serial I/O board.

Since the restart instruction saves the program counter, which is set to the next instruction of the interrupted program, a simple return at the end of the interrupt service routine will allow the program to resume where it left off. Of course, all CPU registers altered by the service routine have to be returned to their state prior to the interrupt. This is best done with PUSH and POP instructions. After the interrupt, the internal flag that enabled the CPU to accept an interrupt, as mentioned above is reset to zero. Thus, the interrupt service routine must reset the flag to 1 using the EI (enable interrupt) instruction. Where this instruction is placed is important. If it is placed at the beginning of the routine, then it is possible that nested interrupts could occur. In other words, an interrupt service routine could be interrupted, which also could be interrupted, and so on. If the EI instruction is placed at the very end of the routine, then a new interrupt will not be serviced until the prior routine is completed. Interrupts must be disabled during CPU controlled data transfers that must be continued to completion. In most CP/M systems, this is during the SECTOR READ or SECTOR WRITE routines when using polled

I/O boards such as the Tarbell single-density controller. If the disk controller has a data buffer (an area to hold data) then interrupts will not cause problems.

Direct Memory Access

The first three I/O methods are commonly termed programmed I/O techniques as distinguished from the fourth, DMA (Direct Memory Access). When S-100 line 74 is pulled low, 8080 pin 13 is pulled high. This causes the CPU to enter a HOLD state. During this time, the CPU relinquishes control of the computer address buss (group of lines used to transfer address information) and data bus. The I/O device then can directly transfer information to and from memory at much greater speed than is possible when the CPU is involved during standard programmed I/O. There are two general ways in which DMA can be implemented. In the first, the calling routine loads certain registers of the device with data such as sector, track, or DMA address, etc., and then issues a data transfer command. In the second method, an area of memory is loaded with this information, and the DMA controller reads this area during a DMA sequence. I will give more information on this in BIOS-PART II. While in a HOLD state, the 8080 CPU will not honor interrupts; thus, interrupt-driven I/O routines will not conflict with DMA routines. An example of a non-disk device that may use DMA is a video board that uses system memory for its refresh memory as is the case with the Processor Technology VDM-1 video board.

Sample Routines

I will now give a general outline on how to implement interrupts in CP/M. The examples that I will use will be based on the MITS 2SIO board set up at port location 10H with interrupt vector 1 (RST 1 will be used). Port 10H is the status port and 11H is the DATA port. Listing 2 shows is a simple output routine using interrupts, as opposed to polled I/O, as demonstrated in the above example. This could drive a printer or a terminal. The main difficulty with output versus input is that the I/O device interrupts the computer as soon as the current operation is completed, and thus could place the system in a infinite loop between the main program and the interrupt service routine. To avoid this problem, the I/O port's output interrupt is disengaged until data is ready to be printed. Since all disk I/O routines must have DI and EI commands, the DI command would not work to disable the interrupts. A way of getting around this would be to use an interrupt status byte in memory, which would be checked at the end of each disk I/O to determine whether interrupts should be enabled using the EI command. If input interrupts are to be allowed, however, then this scheme would not work.

In practice, this output routine is no more efficient than polled I/O. Studying the routine will show that there is a built- in wait period when the computer is ready to output information but the I/O device is not ready. Memory buffers can be written into the I/O routine to greatly improve its efficiency. Listing 3 shows an input routine with a buffer. In this example, the interrupts will be disabled once the buffer is full. Thus, any data input after this state occurs will be lost until the buffer is emptied. Although there are many ways to implement a buffer management scheme, I chose to demonstrate a circular buffer using three counters and two pointers. POINT1 is the location where each byte is stored in the buffer as it is input from the I/O device. POINT2 is the location where data is read from the buffer to the main program. POS1 and POS2 are used to keep track of each pointer's position in the buffer. When the counter reaches zero (the pointer is at the end of the buffer) the pointer is set to the beginning of the buffer, and the counter is loaded with the length of the buffer in bytes. Thus the pointers move through the buffer in an endless circle.

The counter labeled COUNT represents the current amount of data waiting to be read by the computer, and is the space in number of bytes between pointers 1 and 2. When data is input from the I/O device, COUNT is incremented, and when data is read by the main program, it is decremented. COUNT is also used by the INSTAT routine to indicate to a main program when data is ready to be read. Another scheme would be to use comparison routines to indicate buffer placement and status but, in experimentation, I found these to be much longer and less efficient in terms of processing time than the method used here. Finally, listing 3 contains additional routines and was taken directly from a working BIOS that I have been using.

Character Traps

A technique that proves quite useful is character-trapping. By using 8080 CPI (compare immediate) instructions in drivers, certain characters can initiate special routines that will implement a user-defined function. In listing 3, I have placed two character traps that have proven very useful extentions to normal CCP and/or BDOS operation. By pressing the ESC key (Escape - ASCII 1BH) on my terminal, the interrupt service routine branches to a routine that awaits a second character from the terminal. In this example I have implemented just two functions, but an unlimited number could be added (with the amount of memory space available for BIOS being the only restriction). The ESC-P sequence enables output to both the console device and the list device. This is similar to the cntr-P function of CP/M, but in this case printing can be enabled or disabled in all situations and is not disabled automatically after WARM boots (cntr-C). A variation on this would be to enable other printers that have I/O drivers in BIOS. Thus, high-speed dot matrix printers versus slower wordprocessing printers could be switched on or off quite easily during execution of programs such as Microsoft BASIC-80 that allow only one list device. The second function, ESC-C, is virtually the same as cntr-C except that a warm boot can be performed at any time unless BIOS has been altered or interrupts disabled. This function is especially useful when a program "hangs-up" or the user wishes to terminate program execution and contr-C does not work. Since warm boots do not change the TPA (transient program area), ESC-C can be used instead of system reset, which does destroy

the TPA. Thus, ESC-C can be performed, and the TPA can be saved on disk. Care should be used when implementing a function such as this during disk I/O to prevent data loss. Although the ESC-P function may be used in polled I/O routines, ESC-C will not work if I/O does not occur after a program bombs.

A final technique that can be implemented via character traps is character conversion. One interfacing project that my consulting associate and I had was to interface a word processing printer to S-100 hardware and CP/M software. This printer uses escape sequences similar to ones implemented above to initialize a wide variety of features such as underlining, reverse print, and boldface (double strike). In order to initialize these functions, I incorporated a trap in the output routine to convert a printing character such as "}" (ASCII 7DH) to an ESC (ASCII 1BH). In the following example, ESC-A causes the printer to boldface print and ESC-B causes it to print normally.

DISPLAY AT TERIMINAL:
Now is the }Atime}B for all...

OUTPUT TO PRINTER
Now is the (ESC A)time(ESC B) for all...

PRINT-OUT
Now is the time for all...

When writing routines such as these, care must be used in selecting trap characters. Since traps often filter characters out of the data input stream, trapped data must not be important to application programs. For example, ESC is used by the "MICROSOFT Basic-80" line editor.

A final example of character trapping is Listing 4. This partial listing shows how to implement back space character deletion in a 1.4 BIOS. This will emulate the backspace option of CP/M 2.0.

The IOBYTE

When implemented in the BIOS, the Intel standard I/O byte function can be used to convert logical devices to specific physical devices. As an example, the console device could be a CRT terminal, a video board with separate keyboard, a printing terminal with keyboard input, or an acoustic coupler that would allow a remote terminal be the console device over the telephone lines. Although it is the responsibility of the BIOS to direct I/O to a physical device, the STAT utility or an application program can be used to modify device assignments if the BIOS is programmed to handle such functions. Location 3 of system memory is reserved for a software register labeled IOBYTE which indicates which physical device is to be assigned to a logical device. CP/M recognizes four logical devices:

1: CON:	Console	The device used by the CCP
2: LST:	List	The printer or hardcopy device
3: RDR:	Reader	Paper tape reader
4: PUN:	Punch	Paper tape punch

When BDOS directs I/O to one of these logical devices, it does so by calling a location in the BIOS vector jump table described above. The table below shows the link between logical and physical device drivers.

LOGICAL DEVICE	VECTOR JUMP TABLE
1: CON:	3: CONSOLE STATUS
	4: CONSOLE INPUT
	5: CONSOLE OUTPUT
2: LST:	6: LIST OUTPUT
3: RDR:	8: READER
4: PUN:	7: PUNCH

The IOBYTE is divided into four 2-bit segments. Each segment refers to one of the logical devices listed above. Information about physical assignments are placed in each segment in the form of binary numbers. Thus each segment can represent four different assignments. Below is a diagram of the IOBYTE showing the positional relationships of logical status information.

```
*****************************
* LST: * PUN: * RDR: * CON: *
*****************************
bit 7 6 5 4 3 2 1 0
```

The following information is the Intel standard logical to physical conversion for the four numbers stored in each segment. The user may or may not follow this standard in BIOS, but the numbers are stored in this manner when using the STAT utility to modify the IOBYTE.

1. CON:	0. TTY
	1. CRT
	2. BAT - Batch mode
	3. UC1 - User-defined console
2. RDR:	0. TTY
	1. PTR - High-speed reader
	2. UR1 - User-defined reader 1
	3. UR2 - User-defined reader 2
3. PUN:	0. TTY
	1. PTP - High-speed punch
	2. UP1 - User-defined punch
	3. UP2 - User-defined punch
4. LST:	0. TTY
	1. CRT
	2. LPT - Line printer
	3. UL1 - User-defined list

In the above list, CRT corresponds to Cathode Ray Terminal. TTY refers to ASR (automatic send/receive) printing terminals. These terminals generally have a built-in paper tape reader/punch and run at very slow speed (11 characters per second). Thus the paper tape device is referred to as a slow reader or slow punch. In contrast to the TTY punch, the fast reader or punch is a separate device that can run at relatively high speeds. For example, DIGITAL EQUIPMENT'S PC11 reader/punch will read at 300

cps and punch at 50 cps. Line printers are usually devices dedicated to hard copy (in contrast to TTYs) and have speeds that range from 30 cps to beyond 180 cps. I have used 180 cps as a general limit because true line printers which print a wholeline at a time (as opposed to character printers which do one character at a time) are really not practical for small systems. Although fast, line printers are quite large, use vast amounts of power, are expensive, and require a great deal of hardware and software interfacing.

Not all of these devices need be written into a BIOS. The following approach gives the most efficient use of space and processor time when writing IOBYTE-directed routines:

1: Write a set of drivers (input, output, input status) for each physical device to be accessed by the system.
2: Avoid including drivers that are not needed.
3: Write a linkage routine that reads and translates IOBYTE information for each position needed in the vector jump table.
4: If possible place these routines in ROM (Read Only Memory) with its own vector jump table to facilitate programming.

If this plan is followed, redundancy will be eliminated and system generation (as when updating or changing software) will be simplified. Hardware will tend to remain static, whereas software goes through a constant evolutionary process. Listing 5 gives an example of how to write a set of linkage routines for the console device. As a final note, reviewing the sections on STAT.COM and PIP.COM in the CP/M documentation will give further insight into the use of IOBYTE.

The next article in this series will deal with disk functions found in a standard BIOS. Material on CP/M file organization will be given as an extension of information introduced in Article II of this series.

References:

"CP/M 2.0 User's Guide", (Set of 7 Manuals), Digital Research, pacific Grove, Ca.

Epstein, Jake: "An Introduction to CP/M", S-100 Microsystems, vol 1, nos. 1, 2, 3, Jan-Jul 1980.

Findley, Robert: "Scelbi 8080 Software Gourmet Guide & Cook Book", 2nd Ed., Scelbi Computer Consulting, Milford Conn., 1978.

"8080 Assembly Language Programming Manual", Intel Corp., Santa Clara, Ca, 1976.

"8080 Microcomputer Systems User's Manual", Intel Corp., Ibid.

"The IEEE S-100 Proposed Bus Standard", Reprint, S-100 Microsystems, vol. 1, no. 1, Jan/Feb, 1980.

Libes, Sol: "S-100 Bus - New Versus Old", S-100 Microsystems, vol. 1, no. 2, Mar/Apr, 1980.

Nichols, Martin: "An Improved CP/M BIOS For Tarbell Disk Controller", S-100 Microsytems, vol. 1, no. 2, Mar/Apr, 1980.

Osborne, Adam: "An Introduction to Microcomputers", vol2.-"Some Real Products", Adam Osborne and Associates, Berkely, Ca., 1976.

—PROGRAM BEGINS NEXT PAGE—

```
                ;**************
                ;* LISTING 1 *
                ;**************

                ;THIS PROGRAM DEMONSTRATES WRITING CALLS TO BIOS
                ;I/O ROUTINES THAT WILL WORK IN ANY SIZED CP/M SYSTEM
                ;WITHOUT MODIFICATION.

                ;THE TECHNIQUE USED IS TO COMPUTE THE LOCATION OF THE
                ;BIOS VECTOR JUMP TABLE USING THE WARM BOOT ADDRESS
                ;FOUND AT LOCATION 01, AND THEN COMPUTE
                ;THE ACTUAL LOCATION OF JUMP VECTOR USING OFFSETS.


                ;THE FOLLOWING TABLE GIVES THE RELATIVE POSITIONS OF THE
                ;BIOS VECTOR JUMP TABLE BASED ON THE WARM BOOT ADDRESS

0000 =          WARM    EQU     00      ;WARM BOOT
FFFD =          COLD    EQU     WARM-3  ;COLD BOOT ADDRESS
0003 =          CONST   EQU     WARM+3  ;CONSOLE STATUS
0006 =          CONIN   EQU     CONST+3 ;CONSOLE INPUT
0009 =          CONOT   EQU     CONIN+3 ;CONSOLE OUTPUT
000C =          LIST    EQU     CONOT+3 ;LIST
000F =          PUNCH   EQU     LIST+3  ;PAPER TAPE PUNCH
0012 =          READER  EQU     PUNCH+3 ;PAPER TAPE READER
0015 =          HOME    EQU     READER+3;HOME DISK
0018 =          SELECT  EQU     HOME+3  ;SELECT DISK DRIVE
001B =          SETRK   EQU     SELECT+3;SET TRACK
001E =          SETSEC  EQU     SETRK+3 ;SET SECTOR
0021 =          SETDMA  EQU     SETSEC+3;SET DMA ADDRESS
0024 =          READ    EQU     SETDMA+3;READ ON SECTOR
0027 =          WRITE   EQU     READ+3  ;WRITE ONE SECTOR

0001 =          WBOOT   EQU     01      ;LOCATION OF WARM BOOT JUMP
0100 =          TPA     EQU     100H    ;BEGINNING OF TRANSIENT PROGRAM AREA

0100                    ORG     TPA     ;LOCATION IS BEGINNING OF TPA

                ;THIS PROGRAM INPUTS A CHARACTER FROM THE KEYBOARD
                ;THEN SENDS IT TO THE LIST DEVICE. NO PRINTOUT WILL
                ;APPEAR ON THE CRT SCREEN.

                START:

0100 310001             LXI     SP,TPA  ;SET STACK POINTER
0103 CD1201             CALL    INCHR   ;GET A CHARACTER
0106 FE03               CPI     03      ;IS CHARACTER CNTR-C
0108 CA0000             JZ      00      ;IF SO DO A WARM BOOT
010B 4F                 MOV     C,A     ;BIOS EXPECTS CHARACTER IN C
010C CD1A01             CALL    PRNCHR  ;PRINT THE CHARACTER
010F C30301             JMP     START+3 ;LOOP AVOIDING FIRST INSTRUCTION

                ;
                ;ROUTINE TO COMPUTE LOCATION OF INPUT VECTOR JUMP
                ;AND THEN BRANCH TO IT

                INCHR:

0112 2A0100             LHLD    WBOOT   ;LOAD THE H,L REGISTER WITH WARM BOOT
                                        ;ADDRESS IN BIOS VECTOR JUMP TABLE
0115 110600             LXI     D,CONIN ;LOAD D,E WITH POSITION OFFSET
0118 19                 DAD     D       ;COMPUTE LOCATION BY ADDING D,E TO H,L
0119 E9                 PCHL            ;PLACE CONTENTS OF H,L IN PROGRAM
                                        ;COUNTER CAUSING A JUMP

                ;ROUTINE TO COMPUTE LOCATION OF LIST VECTOR JUMP AND
                ;THEN BRANCH TO IT

                PRNCHR:

011A 2A0100             LHLD    WBOOT   ;LOAD H,L WITH WARM BOOT ADDRESS
011D 110C00             LXI     D,LIST  ;LOAD D,E WITH OFFSET
0120 19                 DAD     D       ;COMPUTE ADDRESS
0121 E9                 PCHL            ;BRANCH

0122                    END


                ;**************
                ;* LISTING 2 *
                ;**************

                ;THIS IS A DEMONSTRATION OF A VERY SIMPLE
                ;INTERRUPT DRIVEN OUTPUT ROUTINE. THE MEMORY
                ;LOCATION LABELED INTSTAT IS USED INDICATE
                ;WHETHER OR NOT A CHARACTER HAS BEEN PRINTED
                ;DURING AN INTERRUPT.

0031 =          INTEN   EQU     31H     ;BYTE USED TO INITIALIZE INTERRUPTS
                                        ;ON I/O PORT FOR OUTPUT ONLY
0011 =          INTDIS  EQU     11H     ;DISABLES INTERRUPTS ON PORT
0000 =          CSTAT   EQU     00      ;STATUS PORT
0001 =          CDATA   EQU     01      ;DATA PORT


0008                    ORG     08H     ;LOACATION ENTERED AFTER RST 1

0008 C31401             JMP     INTSRV  ;BRANCH TO INTERRUPT SERVICE ROUTINE

0100                    ORG     100H    ;USED FOR DEMONSTRATION PURPOSES

                CONOT:
0100 3A2701             LDA     INTSTAT ;PUT INTERRUPT STATUS IN A
0103 A7                 ANA     A       ;SET FLAGS
0104 C20001             JNZ     CONOT   ;LOOP IF CHARACTER WAITING FOR
                                        ; INTERRUPT
0107 3C                 INR     A       ;SET A TO 1
0108 322701             STA     INTSTAT ;SAVE CHARACTER READY FOR INTERRUPT
010B 79                 MOV     A,C     ;GET CHARACTER
010C 322801             STA     OUTCHR  ;SAVE CHARACTER
010F 3E31               MVI     A,INTEN ;ENABLE INTERRUPTS AT PORT
0111 D300               OUT     CSTAT   ;INITIALIZE PORT
0113 C9                 RET

                ;OUTPUT INTERRUPT SERVICE ROUTINE
                ;REGISTERS USED ARE SAVED TO PREVENT LOSS
                ;OF DATA NEEDED BY MAIN PROGRAM
                ;NOTE: THIS ROUTINE WOULD BE ENTERED VIA A JUMP
                ;FROM ONE OF THE 8 LOCATIONS BETWEEN 00 AND 38H.
                ;THE LOCATION IS DETERMINED BY THE DATA PLACED
                ;ON THE BUS BY THE I/O PORT HARDWARE DURING THE
                ;INTERRUPT.
```

```
                INTSRV:

0114 F5                 PUSH    PSW     ;SAVE A AND STATUS FLAGS
0115 C5                 PUSH    B       ;SAVE B,C REGISTER PAIR
0116 3A2801             LDA     OUTCHR  ;GET CHARACTER
0119 D301               OUT     CDATA   ;PRINT IT
011B AF                 XRA     A       ;SET A TO 0
011C 322701             STA     INTSTAT ;RESET INTERRUPT STATUS
011F 3E11               MVI     A,INTDIS;GET INITIALIZATION BYTE
0121 D300               OUT     CSTAT   ;INITIALIZE PORT
0123 C1                 POP     B       ;GET DATA BACK
0124 F1                 POP     PSW     ;GET DATA BACK
0125 FB                 EI              ;ENABLE INTERRUPTS
0126 C9                 RET             ;RETURN TO MAIN PROGRAM


0127 00         INTSTAT:DB      00      ;INTERRUPT STATUS
0128 00         OUTCHR: DB      00      ;CHARACTER STORAGE
```

```
;**************
;* LISTING 3 *
;**************

;THIS LISTING DEMONSTRATES INTERRUPT DRIVEN INPUT AND
;OUTPUT. THESE ROUTINES WERE EXTRACTED FROM A WORKING
;BIOS AND ARE ONLY MINIMALY COMMENTED. THE INPUT SERVICE
;ROUTINE ALLOWS THE USER TO EMULATE CP/M CNTR-P OPTION -
;PRINTING ON LIST DEVICE OR PROGRAM TERMINATION - CP/M
;CNTR-C OPTION.

;THE SERIAL BOARD USED IS THE MITS 2SIO WHICH USES THE
;MOTOROLA 6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE.
;THIS CHIP IS INTIALIZED BY OUTPUTING 0B1H FOR INPUT AND
;OUTPUT INTERRUPTS, 031H FOR OUTPUT ALONE, 91H FOR INPUT
;ALONE, AND 11H FOR NO INTERRUPTS (POLLED I/O MODE)


EQU     BUFLEN  10H     ;DETERMINES SIZE OF INPUT BUFFER

;
;INTERUPT SERVICE ROUTINE
;CHECKES CSTAT TO FIND WHICH PORT, IN OR OUT
;INTERUPTED. IF BOTH AT SAME TIME WILL SERVICE
;IN FIRST

SRVINT: PUSH    PSW
        IN      CSTAT   ;GET CONSOLE STATUS
        ANI     01      ;CHECK FOR INPUT READY
        JNZ     INTSRV  ;IF SO GO FOR INPUT
        JMP     OINTSRV ;OTHERWISE GO FOR OUTPUT

; CHECK CONSOLE INPUT STATUS.
;

CONST:  LDA     COUNT   ;GET NUMBER OF CHARACTERS WAITING
                        ; TO BE INPUT
        ANA     A       ;SET FLAGS
CONST1: MVI     A,00    ;SET A = 0
        RZ      ;RETURN IF COUNT=0
        CMA     ;COMPLEMENT A, A=0FFH
        RET
```

```
;
; READ A CHARACTER FROM CONSOLE.
;
CONIN:
        LDA     COUNT
        ANA     A
        JZ      CONIN   ;LOOP TILL INTERRUP
        DCR     A
        STA     COUNT
        LHLD    POINT2  ;PUT POINTER 2 IN H,L
        MOV     B,M     ;GET CHARACTER
        LDA     POS2
        DCR     A
        JZ      CONIN2
        INX     H
CONIN1: STA     POS2
        SHLD    POINT2
        EI
        MOV     A,B
        ANI     7FH
        RET
;
;THIS ROUTINE SET POINTER AND POSITION TO
;BEGINNING OF BUFFER
;

CONIN2: MVI     A,BUFLEN
        LXI     H,BUFFER
        JMP     CONIN1


;INTERUPT SERVICE ROUTINE
;

INTSRV: PUSH    H
        IN      CDATA   ;GET DATA
        ANI     7FH     ;STRIP PARITY BIT
        CPI     1BH     ;ESC - ESCAPE
        JZ      ATTN
        LHLD    POINT1  ;GET POINTER
        MOV     M,A     ;SAVE CHARACTER
        LDA     COUNT   ;GET CHARACTERS
        INR     A       ;UPDATE COUNT
        STA     COUNT
        LDA     POS1    ;GET BUFFER POSITION
        DCR     A       ;DECREMENT POSITION
        JZ      INTSRV2
        INX     H       ;UPDATE POINTER
INTSRV1:STA     POS1
        SHLD    POINT1  ;SAVE IT
INTSRV3:POP     H
        POP     PSW
        EI
        RET

INTSRV2:MVI     A,BUFLEN
        LXI     H,BUFFER
        JMP     INTSRV1
```

```
;ROUTINE TO CHECK FOR SPECIAL CONTROL CHARACTERS

ATTN:   IN      CSTAT
        ANI     01
        JZ      ATTN
        IN      CDATA
        ANI     7FH
        CPI     'C'     ;WARM BOOT
        JZ      00      ;TERMINATE
        CPI     'P'     ;PRINT TOGGLE
        JNZ     INTSRV3
        LDA     PRNECO
        XRI     1       ;TOGGLE BIT 0
        STA     PRNECO
        JMP     INTSRV3

; WRITE A CHARICTER TO THE CONSOLE DEVICE.
;THIS IS EXPERIMENTAL OUTPUT INTERUPTED ROUTINE
;
CONOT:
        LDA     ISTAT
        ANA     A
        JNZ     CONOT
        INR     A
        STA     ISTAT
        MOV     A,C
        STA     OUTCHR
        MVI     A,0B1H
        OUT     CSTAT
        RET

OINTSRV:

        PUSH    B
        LDA     OUTCHR
        OUT     CDATA
        MOV     C,A
        XRA     A
        STA     ISTAT
        MVI     A,091H
        OUT     CSTAT
        LDA     PRNECO  ;GET PRINTER STATUS
        ANI     01
        CNZ     LIST    ;IF PRINTER IS ON, PRINT CHARACTER
        POP     B
        POP     PSW
        EI
        RET

ISTAT:  DB      00
OUTCHR: DB      00
PRNECO: DB      00      ;1=PRINTER ON, 0=OFF

COUNT   DB      00
POINT1  DW      BUFFER
POS1    DB      BUFLEN
POINT2  DW      BUFFER
POS2    DB      BUFLEN
BUFFER  DS      BUFLEN
```

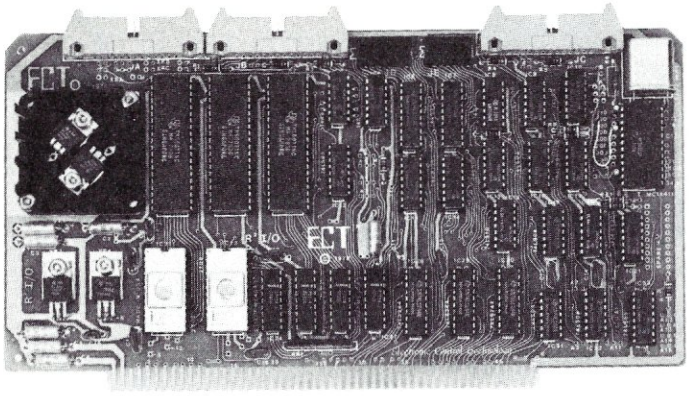
```
;*************
;* LISTING 4 *
;*************

;THIS LISTING SHOWS AN EXAMPLE OF IMPLEMENTING
;CHARACTER DELETION USING BACKSPACE FOR CP/M 1.4
;AND EARLIER SYSTEMS. THE MEMORY LOCATION LABELED
;DELSTAT, DELETION STATUS, IS USED BY THE OUTPUT
;ROUTINE TO DETERMINE WHEN TO ERASE A CHARACTER
;FROM THE CONSOLE SCREEN. THIS PROCEDURE SHOULD ONLY BE
;USED WITH CRT DEVICES AS OPPOSED TO TTY'S.

0010 =          CSTAT   EQU     10H
0011 =          CDATA   EQU     CSTAT+1


0100                    ORG     100H    ;USED FOR DEMONSTATION PURPOSE

0100 DB10       CONIN:  IN      CSTAT   ;GET STATUS
0102 E601               ANI     01      ;CHECK BIT 0
0104 CA0001             JZ      CONIN   ;LOOP TILL READY
0107 DB11               IN      CDATA   ;GET CHARACTER
0109 E67F               ANI     07FH    ;RESET PARITY BIT TO 0
010B FE7F               CPI     7FH     ;IS IT DELETE/RUBOUT
010D C0                 RNZ             ;RETURN IF NOT SO
010E 3EFF               MVI     A,0FFH  ;GET DELETION IN PROGRESS BYTE
0110 323A01             STA     DELSTAT ;SAVE IT
0113 C9                 RET

                CONOT:

0114 DB10               IN      CSTAT   ;GET PORT STATUS
0116 E602               ANI     02      ;SET FLAG
0118 CA1401             JZ      CONOT   ;LOOP TILL PORT READY
011B 3A3A01             LDA     DELSTAT ;GET DELETION STATUS
011E A7                 ANA     A       ;SET FLAGS
011F C22601             JNZ     CONOT1  ;JUMP IF NOT ZERO
0122 79                 MOV     A,C     ;GET CHARACTER
0123 D311               OUT     CDATA   ;PRINT IT
0125 C9                 RET

0126 AF         CONOT1: XRA     A       ;CLEAR A
0127 323A01             STA     DELSTAT ;CLEAR DELETION STATUS
012A 0E08               MVI     C,08    ;GET BACKSPACE
012C CD1401             CALL    CONOT   ;PRINT IT
012F 0E20               MVI     C,20H   ;GET SPACE - WILL CLEAR DELETED
                                        ; CHARACTER FROM SCREEN
0131 CD1401             CALL    CONOT   ;PRINT IT
0134 0E08               MVI     C,08    ;GET BACKSPACE
0136 CD1401             CALL    CONOT   ;PRINT IT
0139 C9                 RET

013A 00         DELSTAT:DB      00      ;FFH=DELETION,00=NO DELETION
013B C9                 RET
```

```
                ;**************
                ;* LISTING 5 *
                ;**************

                ;THIS SKELETON LISTING IS A DEMONSTRATION OF HOW TO
                ;PROGRAM A BIOS WITH IOBYTE IMPLEMENTED.
                ;ALTHOUGH THE ACTUAL DRIVERS ARE NOT LISTED, EACH ROUTINE IS
                ;SUPPLIED SO THAT THE READER CAN GET A FEEL FOR HOW THE SYSTEM
                ;WORKS. THE VARIOUS DEVICES IN A HYPOTHETICAL SYSTEM COULD BE:
                ;
                ;     TTY - AN ASR TYPE TTY
                ;     CRT - 9600 BAUD TERMINAL WITH DETACHABLE KEYBOARD
                ;     BAT - A MODEM BOARD
                ;     UC1 - INPUT - FROM CRT DETACHABLE KEYBOARD
                ;           OUTPUT - VIDEO BOARD FOR HIGH SPEED WORDPROCESSING
                ;     LPT - 180 CPS DOT MATRIX PRINTER
                ;     UL1 - 45 CPS DAISYWHEEL PRINTER FOR WORDPROCESSING

                ;THE LISTING ALSO SHOW HOW TO COMPUTE THE STARTING POINT
                ;OF BIOS GIVEN VERSION AND MEMORY SIZE.



0003 =          IOBYTE  EQU     03      ;IOBYTE IS LOCATED AT LOCATION 3
0014 =          MSIZE   EQU     20      ;USER WOULD FILL THIS IN FOR SYSTEM
1600 =          SYSTEM  EQU     800H+0E00H;LENTH OF 2.0 CCP AND BDOS
0000 =          BIAS    EQU     MSIZE-20;SET UP FOR CP/M 2.0 (SEE ARTICLE
3400 =          TPA     EQU     3400H   ;TPA SIZE IN MINIMAL SYSTEM + 100H

4A00                    ORG     TPA+SYSTEM+(BIAS*400H)



                ;VECTOR JUMP TABLE FOR A 20K SYSTEM

4A00 C3124A             JMP     BOOT    ;COLD BOOT
4A03 C3134A             JMP     WBOOT   ;WARM BOOT
4A06 C3144A             JMP     CONST   ;CONSOLE INPUT STATUS
4A09 C3294A             JMP     CONIN   ;CONSOLE INPUT
4A0C C33E4A             JMP     CONOT   ;CONSOLE OUT
4A0F C3534A             JMP     LIST    ;LIST DEVICE

                ;TABLE CONTINUES ON FROM HERE



                ;COLD BOOT ROUTINE WOULD GO HERE IN ACTUAL LISTING.
                ;RET USED SO THAT ASSEMBLER WILL WORK WITHOUT ERRORS.

4A12 C9         BOOT:   RET             ;FOR DEMO

                ;WARM BOOT ROUTINE WOULD GO HERE IN ACTUAL LISTING

4A13 C9         WBOOT:  RET             ;FOR DEMO
```

```
                ;CONST IS USED TO DETERMINE IF A CHARACTER IS READY
                ;AT THE CONSOLE DEVICE. WHICH CONSOLE DEVICE THAT IS TESTED
                ;IS DETERMINED BY I/O BYTE.

                CONST:

4A14 3A0300             LDA     IOBYTE  ;GET IOBYTE
4A17 E603               ANI     03      ;CHECK CONSOLE SEGMENT
                                        ;  AND STRIP OTHER SEGMENTS
4A19 CA684A             JZ      TTYST   ;CHECK TTY INPUT STATUS
4A1C FE01               CPI     01      ;SET FLAGS
4A1E CA6B4A             JZ      CRTST   ;CHECK CRT INPUT STATUS
4A21 FE02               CPI     02      ;SET FLAGS
4A23 CA6E4A             JZ      BATST   ;CHECK BATCH INPUT STATUS
4A26 C36B4A             JMP     CRTST   ;CHECK USER 1 INPUT STATUS

                ;ROUTINE TO DIRECT PROGRAM TO INPUT ROUTINE
                ;SPECIFIED BY IOBYTE


                CONIN:

4A29 3A0300             LDA     IOBYTE  ;GET IOBYTE
4A2C E603               ANI     03      ;CHECK CONSOLE SEGMENT
                                        ;  AND STRIP OTHER SEGMENTS
4A2E CA694A             JZ      TTYIN   ;INPUT FROM TTY
4A31 FE01               CPI     01      ;SET FLAGS
4A33 CA6C4A             JZ      CRTIN   ;INPUT FROM CRT
4A36 FE02               CPI     02      ;SET FLAGS
4A38 CA6F4A             JZ      BATIN   ;INPUT FROM BATCH DEVICE
4A3B C36C4A             JMP     CRTIN   ;INPUT FOR USER 1 DEVICE


                ;ROUTINE TO DIRECT PROGRAM TO CONSOLE OUTPUT ROUTINE
                ;AS SPECIFIED BY IOBYTE.

                CONOT:

4A3E 3A0300             LDA     IOBYTE  ;GET IOBYTE
4A41 E603               ANI     03      ;CHECK CONSOLE SEGMENT
                                        ;  AND STRIP OFF OTHERS
4A43 CA6A4A             JZ      TTYOT   ;OUTPUT TO TTY
4A46 FE01               CPI     01      ;SET FLAGS
4A48 CA6D4A             JZ      CRTOT   ;OUTPUT TO CRT
4A4B FE02               CPI     02      ;SET FLAGS
4A4D CA704A             JZ      BATOT   ;OUTPUT TO BATCH DEVICE
4A50 C3714A             JMP     UC1OT   ;OUTPUT TO USER 1 DEVICE


                ;ROUTINE TO DIRECT PROGRAM TO LIST DRIVER AS SPECIFIED
                ;BY IOBYTE

                LIST:

4A53 3A0300             LDA     IOBYTE  ;GET IOBYTE
4A56 E6C0               ANI     0C0H    ;CHECK LIST SEGMENT
                                        ;  AND STRIP OFF OTHERS
4A58 CA6A4A             JZ      TTYOT   ;OUTPUT TO TTY
4A5B FE40               CPI     40H     ;SET FLAGS
4A5D CA6D4A             JZ      CRTOT   ;OUTPUT TO TTY
4A60 FE80               CPI     80H     ;SET FLAGS
4A62 CA724A             JZ      LPTOT   ;OUTPUT TO LINE PRINTER
4A65 C3734A             JMP     UL1OT   ;OUTPUT TO USER LIST DEVICE
```

```
                ;ACTUAL DEVICE DRIVERS WOULD BE PLACED HERE. FOR DEMONSTRATION
                ;PURPOSES, RET INSTRUCTIONS ARE USED.
                ;**** TTY ****
                ;ROUTINE TO CHECK INPUT STATUS OF TTY
                TTYST:
4A68 C9                 RET     ;FOR DEMO
                ;ROUTINE TO INPUT FROM TTY
                TTYIN:
4A69 C9                 RET     ;FOR DEMO
                ;ROUTINE TO PRINT TO TTY
                TTYOT:
4A6A C9                 RET     ;FOR DEMO
                ;**** CRT ROUTINES ****
                ;ROUTINE TO CHECK CRT INPUT STATUS
                CRTST:
4A6B C9                 RET     ;FOR DEMO
                ;ROUTINE TO INPUT FROM CRT
                CRTIN:
4A6C C9                 RET     ;FOR DEMO
                ;ROUTINE TO PRINT TO CRT
                CRTOT:
4A6D C9                 RET     ;FOR DEMO
                ;**** BATCH ROUTINES ****
                ;ROUTINE TO CHECK MODEM INPUT STATUS
                BATST:
4A6E C9                 RET     ;FOR DEMO
                ;ROUTINE TO INPUT FROM MODEM
                BATIN:
4A6F C9                 RET     ;FOR DEMO
                ;ROUTINE TO PRINT TO MODEM
                BATOT:
4A70 C9                 RET     ;FOR DEMO
                ;*** USER CONSOLE 1 ****
                ;PRINTOUT TO VIDEO CARD
                UC1OT:
4A71 C9                 RET     ;FOR DEMO
                ;**** LIST DEVICES ****
                ;OUT PUT TO A DOT MATRIX PRINTER
                LPTOT:
4A72 C9                 RET     ;FOR DEMO
                ;OUTPUT TO DAISYWHEEL PRINTER
                UL1OT:
4A73 C9                 RET     ;FOR DEMO
4A74                    END
```

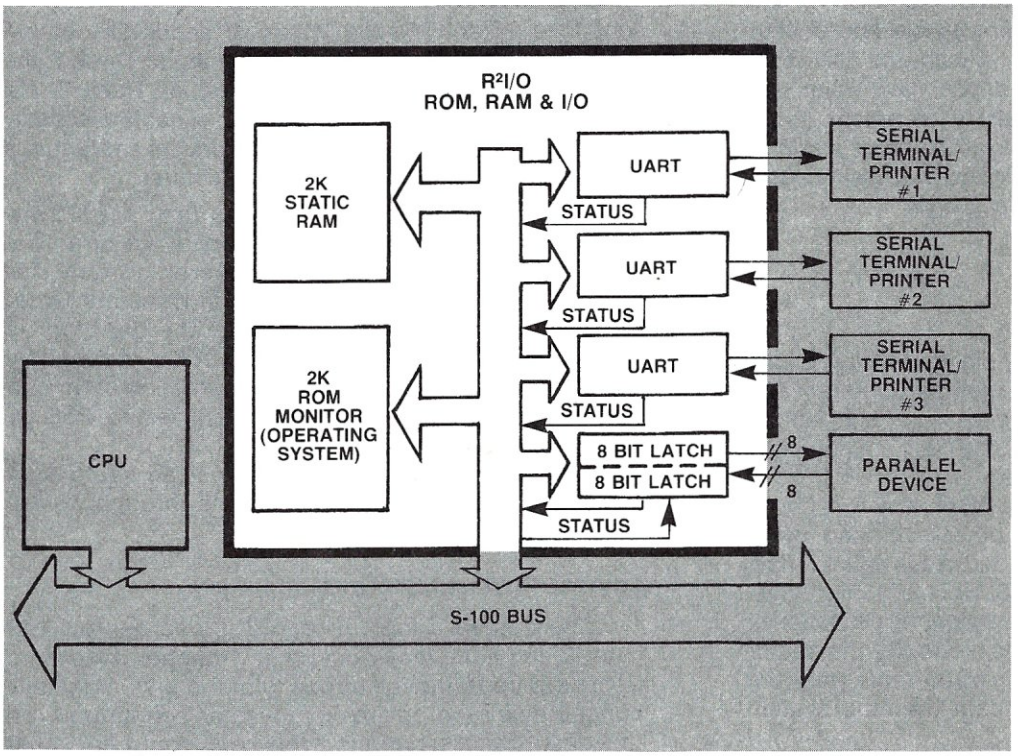
ECT R²I/O... The S-100 ROM, RAM & I/O Board



ELECTRONIC CONTROL TECHNOLOGY's R²I/O is an S-100 Bus I/O Board with 3 Serial I/O Ports (UART's), 1 Parallel I/O Port, 4 Status Ports, 2K of ROM with Monitor Program and 2K of Static RAM. The R²I/O provides a convenient means of interfacing several I/O devices, such as – CRT terminals, line printers, modems or other devices, to an S-100 Bus Microcomputer or dedicated controller. It also provides for convenient Microcomputer system control from a terminal keyboard with the 8080 Apple ROM monitor containing 26 Executive Commands and I/O routines. It can be used in dedicated control applications to produce a system with as few as two boards, since the R²I/O contains ROM, RAM and I/O.

The standard configuration has the Monitor ROM located at F000 Hex with the RAM at F800 Hex and the I/O occupies the first block of 8 ports. Jumper areas provide flexibility to change these locations, within reason, as well as allow the use of ROM's other than the 2708 (e.g. 2716 or similar 24 pin devices). Baud rates are individually selectable from 75 to 9600. Voltage levels of the Serial I/O Ports are RS-232.

- **S-100 BUS**
- **2K ROM**
- **2K RAM**
- **ROM Monitor (Operating System)**
- **3 Serial I/O Ports**
- **1 Parallel I/O Port**
- **4 Status Ports**



8080 APPLE MONITOR COMMANDS

A – Assign I/O
B – Branch to user routine A-Z
C – Undefined
D – Display memory on console in Hex
E – End of file tag for Hex dumps
F – Fill memory with a constant
G – GOTO an address with breakpoints
H – Hex math sum & difference
I – User defined
J – Non-destructive memory test
K – User defined
L – Load a binary format file
M – Move memory block to another address
N – Nulls leader/trailer
O – User defined
P – Put ASCII into memory
Q – Query I/O ports: QI (N)-read I/O; QO(N,V)-send I/O
R – Read a Hex file with checksum
S – Substitute/examine memory in Hex
T – Types the contents of memory in ASCII equivalent
U – Unload memory in Binary format
V – Verify memory block against another memory block
W – Write a checksummed Hex file
X – Examine/modify CPU registers
Y – 'Yes there' search for 'N' Bytes in memory
Z – 'Z END' address of last R/W memory location

Two columns of the last installment of THE CP/M CONNECTION were transposed. The correct order is: On page 32, all of column 2 from the words "and DDT will automatically put..." through page 33 column 1, "ICBIOS.HEX - RA080" should follow the command "A DDT CP/M32.COM" at the bottom of page 33 column 2.

THE CP/M CONNECTION

Part II — CP/M File Operations

by Chris Terry

CP/M is available for a variety of disk drives, controllers and methods, including single and double-density, hard and soft sectoring, 8-inch and 5-1/4-inch disks and Winchester hard disk drives, all of which vary considerably in their disk primitives. In this article, for the sake of simplicity, we consider only the standard distribution version of CP/M Version 1.4, issued on a single-density, soft-sectored, 8-inch disk.

DISK ORGANIZATION

Main Divisions of Disk Space

The standard soft-sectored, single-density, 8-inch disk is divided into 77 Tracks (numbered 0 through 76), and there are 26 Sectors (numbered 1 through 26) per track. This conforms to the IBM 3740 disk layout; such disks are called "IBM-compatible".

Each sector stores 128 data bytes; the two Cyclic Redundancy Check bytes and other overhead bytes which follow the data are not included in this count. Thus, the total storage space is 77*26*128=256,256 bytes. This, too, follows the IBM format, but again is a function of a BDOS table; it is perfectly possible to set the sector size to any multiple of 128 by changing the table entry, but files would not then be portable except to another system with the same blocking factor.

On every disk that runs under CP/M, the storage space is divided into three distinct areas:

- CP/M System Area
- File Directory Area
- File Storage Area

CP/M System Area. Tracks 0 and 1 are always reserved for the CP/M system, although the system need not be present on every disk. The coldstart loader is contained in Track 0 Sector 1; the CCP and BDOS occupy the rest of Track 0 as well as 17 sectors on Track 1; the remaining nine sectors (1152 bytes) of Track 1 are available for the CBIOS. The number of sectors actually used by the CBIOS depends on what drivers and features are included by the controller manufacturer.

File Directory Area. Sixteen sectors on Track 2 are always reserved for the file directory. Each directory entry is 32 bytes long; thus, there is room for (16*128)/32=64 entries in the standard system. Note, however, that sector allocation for the directory is controlled by a table in the BDOS; OEMs licensed by Digital Research Inc. to reconfigure the system can expand the number of directory entries to 255 by changing this table.

File Storage Area. The remaining ten sectors on Track 2 and all sectors on Tracks 3 through 76 are available for files.

Logical/Physical Sector Mapping

The standard logical record is one sector (128 bytes), and a file may occupy any number of sectors from zero up to the full capacity of the disk. Logically consecutive records are not physically contiguous on the disk. This is because the disk controller must process the CRC bytes after reading Logical Sector N, to verify that there were no read errors. Also, the BDOS has some housekeeping chores to perform. If Logical Record N+1 were in fact physically adjacent to Record N, it would probably pass under the read head before the chores were complete; the system would then have to wait until it came round on the next revolution of the disk, about 16 milliseconds later. This would make sequential reading unacceptably slow.

Chris Terry, 324 East 35th St. New York, NY 10016

For this reason, logically consecutive records are mapped onto the disk with several physical sectors between each. The standard skew (sometimes called "interlace") for CP/M is six sectors, to be IBM-compatible, and is shown in Figure 1; this mapping is identical for all directory and file storage tracks. The translation from logical record numbers to physical sector numbers is performed by a lookup table that is usually in BDOS, though some versions put the table in the BIOS. When the table is in the BDOS, disk utilities that use the disk primitives directly must provide a separate translation table of their own. Thus, an application program using the disk primitives to gain access to logical Sectors 19, 20, and 21 of Track 3, would in fact access physical sectors 6, 12, and 18 on that Track. After completing the housekeeping for Sector 6, there is only a minimal wait before Sector 12 (the next in the logical sequence) arrives under the read head.

The routine that performs the logical/physical sector mapping is transparent to the user. In effect this routine says, "Whenever you give me a logical record number, I will convert it to a physical sector number. You don't need to know what that number is, but my mapping will give you quicker access to the data area."

KEEPING TRACK OF DISK SPACE USAGE

Unlike some other microcomputer operating systems, CP/M does not require that the size of a file be specified at the time of creation. Instead, space on the disk is allocated dynamically, as needed. Space that is released as the result of closing a file from which at least 1K has been deleted, can immediately be re-used by another file. The tools that permit this dynamic space allocation are:

- The Allocation Bit Map
- The File Control Block (FCB)
- The Directory

Allocation Bit Map

For every drive configured in the system, the BDOS maintains a space allocation bit map consisting of 243 individual bits. This map is read into memory when the drive is logged in, is modified during Write operations, and is written back to the disk each time a file on that disk is closed. Typing Ctrl-C erases all bit maps from memory except those for Drive A and for the currently logged-in disk.

Each bit in the map represents a group (sometimes called a "cluster") of eight logically consecutive sectors on the disk. The bit positions and their associated groups are numbered 00 through F2 hex (see Figure 2). The first two bits are associated with the first sixteen logical sectors on Track 2. These two groups (00 and 01) contain the file directory, and bits 00 and 01 in the allocation map always contain 1's, even when no directory entries have yet been made. This ensures that the directory can never be overwritten by a file.

```
A > DISKDUMP B:MAP

GROUP ALLOCATION MAP DRIVE -B

                    Columns
      0123456789ABCDEF0123456789ABCDEF
Even 0 1111111111111111|1111111111111111 1
Half- 2 1111111111111111|1111111111111111 3 Odd
rows  4 1111111111111111|1111111111111111 5 Half-
      6 1111111111111111|1110000000000000 7 rows
      8 0000000000000000|0000000000000000 9
      A 0000000000000000|0000000000000000 B
      C 0000000000000000|0000000000000000 D
      E 0000000000000000|00               F

127 GROUPS REMAINING ON DISK OUT OF 243

To find the hex Group number of a bit in the map, take the first digit from the
even or odd half-row and the second digit from the column in the same half.
The first 0 represents 73.
```

Figure 2. Allocation Bit Map

When BDOS receives a request to create a file, it searches the allocation bit map until it finds a bit containing a 0; the number of this bit is the number of the first free group. BDOS then sets the map bit to 1 and places the 1-byte hexadecimal group number in the mapping area of the File Control Block (FCB) created for the new file.

Each time a Write operation is requested for the file, BDOS examines the last group number in the FCB and also the Next Record number, and from these computes the Track and Logical Sector numbers where writing is to take place. When all eight sectors of a group have been filled, BDOS automatically searches the allocation bit map again for the first bit containing a 0. When one is found, its group number is added to the FCB and the map bit is set to 1. Thus, a file which has seven or fewer records will be shown as occupying one group (1K); a file which has eight records will be shown as occupying two groups, even though the second group is empty.

SECTOR ID NUMBERS FORMATTED ON DISK	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
LOGICAL SECTOR #s																										
Read on Pass 1	01					02					03					04						05				
Read on Pass 2				06				07					08					09								
Read on Pass 3			10				11					12					13									
Read on Pass 4		14				15					16					17										18
Read on Pass 5					19					20					21						22					
Read on Pass 6				23					24					25						26						

Figure 1. Standard 6-Sector Skew in Logical/Physical Sector Mapping.

CP/M Connection cont'd...

This process has several important results:

- The minimum space that can be occupied by any file (even an empty file) is eight sectors (1K).
- Because BDOS always searches the allocation bit map from the beginning on a Write request and allocates the first free group it finds, logically consecutive components of a file may be physically located anywhere on the disk and not necessarily in Track/Sector order.
- A Write request will never be denied until the disk is too full to hold the amount of data to be written. Of course, denial of a Write request is a fatal error unless the application program makes provision for mounting a fresh disk in such circumstances, but it very seldom occurs if reasonable care is taken. Use the STAT utility to check available space before undertaking any operation that creates backup or temporary files.
- Disk space is efficiently used. In other systems that require file size to be specified, overcaution can result in large amounts of unused space that is not available to other files. This can only be recovered by copying the data to a new file with the proper size specification. The same procedure has to be followed if it is desired to expand a file that has already used the space originally allocated to it.

File Control Block (FCB)

An FCB is a 33-byte block of read/write memory containing all the information needed by BDOS to find a file on the disk and to access any specified record. Whenever a new file is created, an FCB must be created for it. The area from 005C to 007C hex is the default FCB area used by the CCP; it may also be used by transient programs. If a transient program requires more than one file to be open at the same time, the program must create an FCB for each file that is to be accessed. These FCBs should be in the TPA.

FCB Layout. The layout of an FCB is shown in Figure 3. When a file is first created, the CCP or user program first clears all bytes of the FCB to zero, and then initializes the first thirteen bytes as follows.

ET. Byte 0. The CP/M Manual defines this as "Entry Type, not currently used but assumed zero." While in the FCB area, this byte remains 0.

FN. Bytes 1 thru 8. The CCP or user program places the filename in this field, left-justified. If the name has fewer than eight characters, the remaining bytes are padded with ASCII blanks (20 hex).

FT. Bytes 9 thru 11. The CCP or user program places the 3-character file type in this field. Note that the period which separates filename and type in a command is only a delimiter and is not put in the FCB. If the file type has fewer than three characters, the remaining bytes of the FT field are padded with ASCII blanks. If the file is a temporary file, this field will contain '$$$'.

EX. Byte 12. This byte, initialized to zero, indicates the file extent number. As we shall see, an FCB describes a file segment up to 16K in size, i.e., 128 records (sectors). When 128 sectors have been written, bytes 0 thru 31 of the FCB are copied to the first free slot in the directory area, and the Extent number in the FCB is incremented. Thus, we shall find a separate directory entry, each containing a different extent number, for every 16K segment of a large file.

Bytes 13 and 14 are not used and should always be zero.

RC. Byte 15. This byte, initialized to 0, contains the current number of records in the Extent described by the FCB. As new records are written to the disk, this count is updated by BDOS. Transition of this count from 7F to 80 is the signal for BDOS to copy the FCB to the directory area of the disk and to create a new FCB with the EX and NR bytes updated.

DM. Bytes 16 thru 31. This is the Disk Map area, and is initialized to zeros. When a file is being built, the first Write request causes BDOS to insert the number of the group allocated into Byte 16. No further updating takes place in this field until all sectors of the group have been written. Then BDOS allocates another group and inserts its number into Byte 17, and so on, until all 16 groups (128 sectors) have been written.

IMPORTANT NOTE: Because the FCB is not written to the disk directory area until either 128 records have been written or the file is closed, a system crash can cause the apparent loss of up to 128 records. The data is on the disk but is not recorded in the directory. In applications that entail much data entry, it is good practice to close the file frequently and re-open it; this can avoid painful reconstruction of the directory and the possible destruction of vital data as the result of overwriting from other files after a crash.

NR. Byte 32. This byte, initialized to zero, is updated by BDOS during sequential file operations, and shows the number of the next record to be read or written. For random access, the transient program must place the number of the record to be accessed in this byte before issuing the function call to BDOS. Note that this byte is not copied to the directory entry; it is meaningful only when the file has been opened.

FCB Location. Before we go on to discuss the directory, it is important to emphasize that there is no restriction on the location of an FCB. The CCP and DDT use the area from 005C to 007C hex; this is known as the default FCB area, and is usually given the symbolic label TFCB. When a large file has more than one 16K extent, sequential write operations build the data for each extent in the default FCB area. Sequential read operations cause each directory entry for the file to be fetched into TFCB, in turn. However, a user program may allocate enough memory to hold all the FCBs of a file simultaneously, passing the address of the appropriate FCB to BDOS as one argument of each access request. This will be explained in more detail when we discuss file access operations. Much time and head movement can be saved during random read operations if all of the FCBs for a file are available in RAM, so that they do not have to be fetched from the disk each time a new extent is accessed.

File Directory

The BDOS maintains a directory for each disk. Upon booting CP/M, the contents of Groups 01 and 02

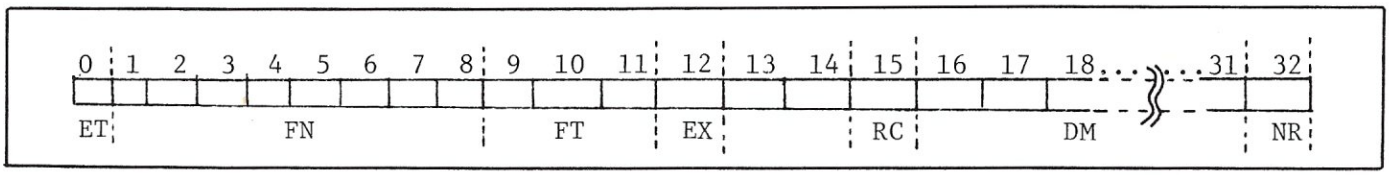


Figure 3. Layout of File Control Block

(16 sectors) are read sequentially from the logged-in disk, and the bit map for that drive is recalculated from the DM bytes of each entry. A request to open a file causes the appropriate directory entry to be copied into the FCB area. If the file has been opened for writing, closing the file causes BDOS to copy the FCB back to the directory area and to rewrite all 16 sectors to the disk immediately.

Directory Layout. A set of typical directory entries is shown in Figure 4. They are for the same disk as the Allocation Map in figure 3. Note that each entry occupies 32 bytes in two lines. The first line (3rd address digit always even) contains 00 in the first byte (to indicate that the entry is for a valid file), followed by the file name and type. The extent number is in byte nnnC, and the record count in nnnF. The second line (3rd address digit always odd) of the entry contains the numbers of the 8-sector groups allocated to the file.

Restoring Erased or Crashed Files. An understanding of this layout may help to recover a file after a system crash or one that has been accidentally erased. The CP/M ERAse command and the Basic KILL command do not in any way modify the disk file. They merely issue a Delete request to BDOS which places E5 in the first byte of the directory entry to mark it as deleted. BDOS also scans the group allocations in line 2 of the entry and sets the corresponding bits in the Allocation Map to 0, thereby freeing these groups for re-use. The file data on the disk remains intact until a Write request to BDOS finds one of these groups free and overwrites one or more sectors on the disk. The disk Dump utility by S.J. Singer on Volume 24 of the CP/M User's Group library can not only read any sector of the disk into an accessible area of memory, but allows examination and replacement of individual bytes before writing the sector back to disk. This facility can restore an erased file to activity by changing the E5 in the directory entry back to 00, provided that the directory entry is intact and that write requests since the erasure have not overwritten the file data on the disk. Restoration also requires Opening the file, to bring the edited directory entry into the FCB area, and then Closing it to restore the Allocation Map and to rewrite this and the restored directory back to the disk.

In the case where the system crashed while a file was open, this disk DUMP utility can also be used to search the file area of the disk for the lost data. If the first extent is preserved in the directory, exploration of the last groups used and the first groups ostensibly unused can provide the first clues on where to look. If two files are open simultaneously for writing, BDOS usually (though not always) allocates open groups alternately to these two files; this, too, can be a help in reconstructing directory entries.

The moral is, never, NEVER allow your application programs to keep writing and writing to a file. If you close and re-open the file every time you write a record, you will never lose any data. If this is too hard on the disk drive, close and re-open the file after every 4K has been written (or some other reasonable amount).

Directory Area Usage. It is worth noting that directory entries on the disk remain intact, even though marked as deleted, until they are overwritten by new entries. Thus, when the directory is brought into memory from the disk, it may contain entries marked for deletion which BDOS will re-use as needed. When BDOS copies an FCB to the directory area, it puts the FCB into the first entry slot that contains E5 in its first byte. As a result, entries for several extents of the same file are not necessarily adjacent, nor even in numerical order. The CP/M DIRectory command and the WDIR (Wide Directory) utility display file names in the order in which they occur in the directory.

If it is desired to alphabetize the directory and to purge entries for deleted files, the SAP (Sort and Pack) utility by Bruce Ratoff (CPMUG Volume 19) can be used. This utility reads the directory from the disk, copying only the active entries into an empty 2K buffer. It then sorts the selected entries into alpha-numeric filename-type-extent order, fills the rest of the buffer with E5, and then writes the sorted and purged directory back to the disk. BE CAREFUL, however. Early versions of SAP operate ONLY on Drive A and are constructed for a particular system size. And, to the best of my belief, existing versions of SAP do not work on double-density systems. Be sure to check the source code, and try it on a backup disk first.

User Program File Access Procedures

When a file-related console command is given (SAVE, ERA, REN, DIR, TYPE) with a drive, filename, and file type, or when a CP/M utility (STAT, PIP, ASM, DDT, etc.) is invoked with a filename as its argument, the CCP or the utility perform all functions required to access the named file, including the creation and updating of the FCB and directory entry. We are here concerned only with the procedures that must be performed by user programs to create, modify, or read data files. Some general principles are explained first; these are followed by some concrete examples.

BDOS Function Calls

CP/M provides 27 different functions, all of which are available to user programs. Functions 1 through 11 relate to peripheral I/O, and are discussed elsewhere. Functions 12 through 27 are disk I/O functions. Only those concerned with creating a new file, reading

CP/M Connection cont'd...

from or writing to an existing or newly created file, or deleting a file are discussed here.

A BDOS function call (that is, a request to BDOS to perform some function) always consists of three operations:

- Load Register C with the function number of the desired operation.
- Load Register Pair DE with the address of the FCB for the file to be accessed. For function 26, load the address of the buffer to be used for disk reads and writes.
- CALL BDOS (entry point is 0005H).

Some, though not all, functions return a result. Single-byte results are returned in the A register. Double-byte results are returned with the low byte in the A register and the high byte in the B register. It is the responsibility of the user program to interpret and use any results returned by BDOS. NOTE: BDOS uses all the registers. If any register values have to be preserved, save them before the BDOS function call and restore them when the function is complete.

Log-In Disk (Function 14)

If the file to be accessed is not on the same disk as the user program, the drive on which the file is (or will be) stored must be logged in. That is, its Allocation Map must be reconstructed in memory before any access can be attempted. Put function number 14 (0EH) in the C register, clear the D register, and load E with the drive number to be logged in. Then call BDOS. No results are returned.

If your program calls for a change of disk, it MUST call for a log-in; if it does not, BDOS will attempt to use the allocation bit map left over from the previous disk on that drive, and existing data on the new disk may be overwritten and permanently lost.

Create a New File

First, allocate space for an FCB. If no other file is open, the TFCB at 005CH may be used; if that is already in use, allocate FCB space (33 bytes) in the transient program area (TPA). Then move the filename (8 characters, left-justified, padded as necessary with ASCII blanks) and the file type (3 characters, left-justified, padded if necessary) into bytes 1 through 11 of the FCB (byte 0 must contain zero).

Load Register C with function number 22 (16H, Make File), load Register Pair DE with the address of the FCB, and call BDOS. BDOS returns the byte address of the directory entry allocated to the file (i.e., an address in the range 00 through 7FH that is relative to the start of the sector in which the entry will be stored on the disk). If the directory is already full, BDOS returns 0FFH in the A register; the user program must check for this and take appropriate action if the directory is full. One possible course would be to print an error message instructing the operator to dismount the current disk and mount a blank formatted disk on the same drive. Upon receiving confirmation via the console that a new disk is mounted, start the operation over by logging in the disk (Function 14) and repeating the Make.

Opening an Existing File

If the file to be accessed already exists, it must be opened before reading or writing can take place. If the file is not on the same disk as the user program, it must be mounted and logged in as described above; if it is on the same disk, the log-in was done when the user program was called.

To open the file, do the following:

- Allocate space for the FCB.
- Move the filename and type into bytes 1 through 11 of the FCB as described for a Make, above.
- Load the C register with function number 15 (0FH, Open File). Load Register Pair DE with the with the address of the FCB.
- CALL BDOS (Entry point is 0005H).
- Clear register A and wait for the completion code to be returned. BDOS returns the byte address of the directory entry if the file is successfully opened, or 0FFH if the file cannot be found.

NOTE: Successful opening of a file says nothing about the mode in which it can be accessed. It merely indicates that the file exists and that its directory entry has been copied into the FCB area. The user program may either read from or write to the file, sequentially or randomly. If only Read operations are performed, the file need not be closed later (although it is good practice to do so). If any kind of Write operation is performed, the file MUST be closed later, in order to ensure that the added or modified space allocations are permanently recorded on the disk, both in the allocation bit map and in the directory.

Buffer Addressing

The starting address of the buffer from which data is to be written to disk, or into which data is to be read from disk, is called the DMA (Direct Memory Access) address. The minimum size of the buffer is 128 bytes, corresponding to one complete sector. Increases of buffer size must be in multiples of 128. The term "DMA" is not strictly accurate unless the controller contains DMA hardware that pre-empts the data bus and transfers a specified number of bytes (starting at the DMA address) at high speed, without intervention of the CPU. However, the term is convenient and has become standard in CP/M.

Unless otherwise specified by a user program, BDOS assumes that all data transfers will take place via the 128-byte buffer at locations 0080H through 00FFH. This is called the Default Buffer, and the standard name of its starting address is TBUFF. This buffer is also used by the CCP for string input from and output to the console.

A user program can change the disk buffer address with a Set DMA function call to BDOS. The procedure is:

- Load Register C with the function code (26=1AH=Set DMA).

THE NEXT GENERATION OF MICROCOMPUTERS IS HERE AT QUASAR DATA PRODUCTS

16 BIT POWER

Z-8000

AND STILL RUN YOUR 8 BIT SOFTWARE

Z-8000 SERIES 16 BIT CPU S-100 BOARD — CAN BE PLUGGED INTO YOUR EXISTING SYSTEM

- FULLY S-100 IEEE COMPATIBLE
- SUPPORTS EXISTING 8 BIT MEMORY AND 8 BIT PERIPHERAL BOARDS
- CAPABLE OF READING AND/OR WRITING 8 BIT, 16 BIT OR MIXED 8 BIT AND 16 BIT MEMORIES AUTOMATICALLY
- 8 BIT AND/OR 16 BIT PERIPHERAL MODULES CAN SIMULTANEOUSLY CO-EXIST IN THE SAME BUS WITHOUT ANY MODIFICATIONS
- CAPABLE OF OPERATING AS A SLAVE PROCESSOR TO ENABLE YOUR EXISTING CPU TO CONTROL THE Z-8000
- SUPPORTS ON-BOARD HARDWARE SINGLE STEPPING
- SUPPORTS EITHER SEGMENTED CPU OR NON-SEGMENTED CPU
- POWER-ON AND RESET JUMP DIP SWITCH SELECTABLE
- JUMPER SELECTABLE 2 OR 4 MHz. OPERATION
- DIP SWITCH SELECTABLE NUMBER AND TYPE OF WAIT STATES
- SOFTWARE
  - Z-80 EMULATOR ENABLES YOU TO EXECUTE YOUR EXISTING 8 BIT SOFTWARE **WITHOUT** ANY MODIFICATIONS AND ALLOWS YOU TO RUN CP/M IMMEDIATELY
  - EXTENDED MONITOR, DEBUGGER, DISASSEMBLER

INDUSTRIAL QUALITY

Z-80

Z-80 SERIES 8 BIT CPU S-100 BOARD

- 4 MHZ. Z-80 MICROPROCESSOR
- FLOPPY DISK CONTROLLER ALLOWING SINGLE- AND DOUBLE-DENSITY USING EITHER 8" or 5¼" DISK DRIVES
- ROOM FOR THREE 2716 EPROMs or THREE 2316 ROMs
- TWO RS-232C SERIAL PORTS
- TWO PARALLEL PORTS
- HARD DISK CONTROL VIA THE PARALLEL PORTS
- REAL TIME CLOCK
- RESET-JUMP CIRCUIT ALLOWS CPU TO JUMP TO MONITOR SOFTWARE ON RESET
- CAPABLE OF PROGRAMMING AND VERIFYING INTEL 2716 ON BOARD WITH EXTERNAL POWER SUPPLY
- UTILIZES VECTORED INTERRUPTS OF Z-80 CPU
- MONITOR PROVIDED

INDUSTRIAL QUALITY

SYSTEMS

QDP-8100 WITH 2 MEGABYTES STORAGE STANDARD (OPTIONAL 4 MEGABYTES)

- Z-8000 SERIES 16 BIT CPU S-100 BOARD - SEE ABOVE
- SOFTWARE (PROVIDED WITH SYSTEM)
  - CP/M 2.2[1] OPERATING SYSTEM
  - BASIC
  - Z80/8080 EMULATOR
  - MONITOR, DEBUGGER, DISASSEMBLER
  - SOFTWARE OPTIONS: PASCAL
  - UNIX[2] OPERATING SYSTEM COMING

QDP-100 WITH 2 MEGABYTES STORAGE STANDARD (OPTIONAL 4 MEGABYTES)

- Z-80 SERIES 8 BIT CPU S-100 BOARD - SEE ABOVE
- SOFTWARE (PROVIDED WITH SYSTEM)
  - CP/M 2.2[1] OPERATING SYSTEM
  - BASIC
  - ACCOUNTS RECEIVABLE, GENERAL LEDGER, ACCOUNTS PAYABLE, PAYROLL WITH COST ACCOUNTING
  - OPTIONAL SOFTWARE: FORTRAN, PASCAL, COBOL, C

EACH SYSTEM CONTAINS:

- INTELLIGENT CRT TERMINAL (80 CHARACTERS X 24 LINES)
- 64 KBYTES RAM
- TWO 8 INCH, DOUBLE SIDED, DOUBLE DENSITY FLOPPY DISK DRIVES WITH CONTROLLER
- 2 SERIAL AND 1 PARALLEL (2 PARALLEL FOR QDP-100) PORTS
- ATTRACTIVE WOODGRAIN CABINET WITH POWER SUPPLIES AND CABLING

FULL TECHNICAL SUPPORT FROM THE STAFF AT QUASAR DATA PRODUCTS

[1]CP/M™ DIGITAL RESEARCH
[2]UNIX™ BELL LABS

---

4 Mhz 64K Dynamic RAM

16K - $250.00 32K - $350.00 48K - $450.00 64K - $549.00

TELETEK DBL. DENSITY, DBL. SIDED

Disk Controller Board..... $395.00

QUASAR FLOPPY SYSTEM

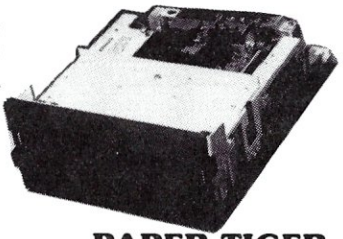
- Two MFE DBL sided drives • Cable • Case & Power Supply assembled and tested Wood cabinet $1895.00

QUASAR 2 MEG FLOPPY

- 2 MFE double sided drives
- Teletek disk controller board
- Power supply & cable
- Wood cabinet
- CP/M version 2.2 & bios
- Assembled & tested $2295.00

Dealer Inquiries Invited, Hours:9-5:30 M-F

Specifications Subject To Change



PAPER TIGER

Includes Graphics $949.00
Cable for TRS-80 ..... $39.00
Call for Apple

MFE Double Sided - Double Density 8" Floppy Disk Drives. (the best) $650.00
Using the Teletek Controller under CP/M, THIS DRIVE WILL GIVE YOU ALMOST ONE MEGABYTE PER DISK DRIVE.
Power supply for above $110.00

TI - 820

Serial Printer -
Full package options... $1995.00

UnixTM - Bell Lab 30 Day ARO CP/MTM - Digital Research

Checks, money orders accepted
Add $2.50 freight charges on orders under 10 lbs. Over 10 lbs. F.O.B. Cleveland

QUASAR DATA PRODUCTS

25151 Mitchell Dr., No.Olmsted, Ohio 44070 (216)779-9387

VISA

- Load the DE Register pair with the starting address of a user buffer.
- CALL BDOS (Entry point at 0005H).

BDOS does not return anything.

Reading and Writing

After a file has been opened, a separate Read or Write request must be issued for each and every sector transferred. Unless otherwise specified by the user program, BDOS assumes that all transfers will take place via the Default Buffer starting at TBUFF (0080H). The user program is responsible for emptying the buffer after each Read (or processing the data while it is still in the buffer), and for filling the buffer before each Write. The request procedure is:

- Load the C Register with the function code (20=15H=Read Next Sector; 21=16H=Write Next Sector.
- Load Register Pair DE with the address of the FCB for the file to be accessed.
- CALL BDOS (entry point is 0005H).

Upon completion of the sector Read or Write, BDOS increments the count in the NR (Next Record) field of the FCB, and returns a completion code in Register A. The completion codes are:

CODE	ON READ	ON WRITE
0	Successful Read	Successful Write
1	Read past EOF	Error in extending the file
2	Sector accessed	End of disk data had no data
255		No more directory space for a new extent

It is the user program's responsibility to check the completion code and to take appropriate action on error conditions. Use of the 128-byte Default Buffer is convenient when the user program must process small quantities of data (for example, a line of source code) before requesting or outputting another record.

There are many occasions when a large block of data must be read into or written from memory in one operation. Examples are reading a .COM file into memory prior to execution, reading a complete set of records that are to be sorted, or making a large block of ASCII text available to speed up string search/replace procedures. To read a large block, do the following:

- Allocate user buffer space, sized to some multiple of 128 bytes.
- Issue a Set DMA request pointing to the start of the user buffer. Store the current DMA address in scratchpad memory.
- Issue the first Read request.
- After each Read request, check the completion code returned by BDOS, and take appropriate action if an error is indicated. Also check for a Buffer Full condition (such as number of sectors read equal to buffer size in sectors).
- After each successful read, get the current DMA address, add 128 to it, and store the updated address in the DE register pair and in the scratchpad. Then issue a new Set DMA request followed by a Read request.

Random Reading

Once a file has been opened, random access to any record is possible by placing the desired record number in byte 31 of the FCB before issuing a Read or Write request (remember that FCB bytes are numbered from 0). Some computation is required here, first to derive the logical record number from the block number (if blocks are larger than 128 bytes), and then to find what extent this record is in.

Suppose that our logical records are 256 bytes (2 sectors) long, and we wish to access record 134. The data we want is in the two sectors starting at 134*2=268 in the sector sequence. However, since a file extent can hold only 128 sectors, sector 268 must be in the third extent. Extent numbers start at 0, so this will be numbered 02. The number of sector 268 relative to the start of Extent 02 is found by taking the remainder of 268 modulo 128; that is, 268%2=2, and the remainder is 268-(128*2)=12 decimal. Since Next Record counts in the FCB also start at 0, the required sector number is 11 (0BH).

Thus, before issuing the access request we must fetch the directory entry for Extent 02 into the FCB area, and place 0BH in the NR field (Byte 31). Now we issue a Set DMA request pointing to the start of a 256-byte buffer, followed by a Read request for the first half of our record. To obtain the second half, we must add 128 to the DMA address and issue a new Set DMA request. We do not need to change the NR field of the FCB because this was incremented automatically by BDOS after the first read, so we finish the operation merely by issuing the second Read request.

Random Writing

Some care must be taken when writing randomly. If we wish to write record 129, for example, we must first have created space for records 1 through 128. We can write 128 records containing nuls, and then add record 129 to the end of these; however, this may be wasteful of disk space, and we could run into trouble if we attempt to write record 2001 (or some high number). Most data management systems use a special CREATE program to create a file of finite size, and then an UPDATE program that enters data into this file in a manner that makes efficient use of the space. There have been a number of articles during the last year on hashing techniques, tree techniques, and indexed sequential access methods. Consult these for further details, which are outside the scope of this article.

Closing a File

It is not necessary to close a file if ONLY read operations were performed on it. This is because reading alone does not change either the Allocation Map or the directory entries for the file. Closure is highly desirable, however, to maintain upward compatibility of the user program with revisions later than 1.4 of CP/M. In a multi-user system, for example, the file would have to be closed before any other user program could access it.

CP/M Connection cont'd...

Further, if any type of writing was done, the space allocations for the file were probably changed, and must be written back to the disk to ensure integrity of the data. The Close function copies the current FCB to the matching directory entry, if one exists, or to the first free slot in the directory area if the current FCB describes a new Extent. Then the allocation bit map is written back to the disk area on which it resides, and the entire updated directory is written out to the first sixteen sectors of Track 2.

To close a file, do the following:

- Load register C with the function code (16=10H=Close).
- Load Register Pair DE with the address of the FCB for the file.
- CALL BDOS (entry point is at 005H).

```
A>DISKDUMP B:C 0-1

                    DRIVE B - TRACK 2  SECTOR 1
0000  00 2D 57 4F 52 4B 20 20  20 30 30 32 00 00 00 01  .-WORK     002....
0010  45 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  E....... ........
0020  00 44 4F 57 48 49 4C 45  53 4C 49 42 00 74 00 07  .DOWHILE SLIB.t..
0030  26 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  &....... ........
0040  00 44 53 4B 44 55 4D 50  31 41 53 4D 00 00 00 80  .DSKDUMP 1ASM....
0050  29 2A 2B 2C 2D 2E 2F 30  31 32 33 34 35 36 37 38  )*+,-./0 12345678
0060  00 44 53 4B 44 55 4D 50  31 41 53 4D 01 00 00 13  .DSKDUMP 1ASM....
0070  39 3A 3B 00 00 00 00 00  00 00 00 00 00 00 00 00  9:;..... ........

                    DRIVE B - TRACK 2  SECTOR 7
0080  00 44 53 4B 44 55 4D 50  31 48 45 58 00 00 00 43  .DSKDUMP 1HEX...C
0090  56 57 58 59 5A 5B 62 65  6A 00 00 00 00 00 00 00  VWXYZ[be j.......
00A0  00 44 53 4B 44 55 4D 50  31 50 52 4E 00 00 00 80  .DSKDUMP 1PRN....
00B0  50 51 52 53 54 55 5C 5D  5E 5F 60 61 63 64 66 67  PQRSTU\] ^ `acdfg
00C0  00 44 53 4B 44 55 4D 50  31 50 52 4E 01 00 00 15  .DSKDUMP 1PRN....
00D0  68 69 6B 00 00 00 00 00  00 00 00 00 00 00 00 00  hik..... ........
00E0  00 44 53 4B 44 55 4D 50  31 53 59 4D 00 00 00 0A  .DSKDUMP 1SYM....
00F0  6C 6D 00 00 00 00 00 00  00 00 00 00 00 00 00 00  lm...... ........

                    DRIVE B - TRACK 2  SECTOR 13
0100  00 4C 4F 41 44 20 20 20  20 43 4F 4D 00 00 00 0E  .LOAD      COM....
0110  6E 6F 00 00 00 00 00 00  00 00 00 00 00 00 00 00  no...... ........
0120  00 4D 41 43 20 20 20 20  20 43 4F 4D 00 00 00 5C  .MAC       COM...\
0130  02 03 04 05 06 07 08 09  0A 0B 0C 0D 00 00 00 00  ........ ........
0140  00 4D 41 43 52 4F 20 20  20 4C 49 42 00 00 00 80  .MACRO     LIB....
0150  3C 3D 3E 3F 40 41 42 46  47 48 49 4A 4B 4C 4D 4E  <=>?@ABF GHIJKLMN
0160  00 4D 41 43 52 4F 20 20  20 4C 49 42 01 00 00 08  .MACRO     LIB....
0170  4F 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  O....... ........

                    DRIVE B - TRACK 2  SECTOR 19
0180  00 4E 43 4E 4D 50 41 52  45 4C 49 42 00 12 00 0A  .NCOMPAR ELIB....
0190  18 19 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
01A0  00 50 49 50 20 20 20 20  20 43 4F 4D 00 00 00 37  .PIP       COM...7
01B0  0E 0F 10 11 12 13 14 00  00 00 00 00 00 00 00 00  ........ ........
01C0  00 53 41 50 20 20 20 20  20 43 4F 4D 00 00 00 04  .SAP       COM....
01D0  15 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
01E0  00 53 45 4C 45 43 54 53  20 4C 49 42 00 69 00 0F  .SELECTS  LIB.i..
01F0  27 28 00 00 00 00 00 00  00 00 00 00 00 00 00 00  '(...... ........

                    DRIVE B - TRACK 2  SECTOR 25
0200  00 53 45 51 49 4F 20 20  20 4C 49 42 00 00 00 52  .SEQIO     LIB...R
0210  1A 1B 1C 1D 1E 1F 20 21  22 23 24 00 00 00 00 00  ...... ! "#$.....
0220  00 53 59 4D 53 54 41 43  4B 4C 49 42 00 44 00 05  .SYMSTAC KLIB.D..
0230  17 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........ ........
0240  00 55 43 41 54 20 20 20  20 53 59 4D 00 00 00 0B  .UCAT     SYM....
0250  43 44 00 00 00 00 00 00  00 00 00 00 00 00 00 00  CD...... ........
0260  00 57 44 49 52 20 20 20  20 43 4F 4D 00 00 00 02  .WDIR      COM....
0270  70 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  p....... ........

                    DRIVE B - TRACK 2  SECTOR 5
0280  00 57 48 45 4E 53 20 20  20 4C 49 42 00 3B 00 06  .WHENS     LIB.;..
0290  25 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  %....... ........
02A0  00 44 53 4B 44 55 4D 50  31 43 4F 4D 00 00 00 18  .DSKDUMP 1COM....
02B0  16 71 72 00 00 00 00 00  00 00 00 00 00 00 00 00  .qr..... ........
02C0  E5 E5 E5 E5 E5 E5 E5 E5  E5 E5 E5 E5 E5 E5 E5 E5  eeeeeeee eeeeeeee
02D0  E5 E5 E5 E5 E5 E5 E5 E5  E5 E5 E5 E5 E5 E5 E5 E5  eeeeeeee eeeeeeee
02E0  E5 E5 E5 E5 E5 E5 E5 E5  E5 E5 E5 E5 E5 E5 E5 E5  eeeeeeee eeeeeeee
02F0  E5 E5 E5 E5 E5 E5 E5 E5  E5 E5 E5 E5 E5 E5 E5 E5  eeeeeeee eeeeeeee
```

Figure 4. Part of Typical Directory

End-of-File Detection

Files of type .COM and .HEX do not contain any built-in end-of-file (EOF) marker. If BDOS starts to process a Read request and finds that the count in the NR field of the FCB is greater than the count in the RC field, the request is aborted and a completion code of 1 (read past end of file) is returned. A transient program may use this indication to break out of a data transfer loop; more usually, such a loop is initialized to read only the number of sectors specified by the RC field.

ASCII files of types ASM, TXT, DOC, etc. can also use the above methods. However, the transient programs that process such files (assemblers, editor, text formatters, etc.) expect to find at least one ctrl-Z (1AH) code after the CRLF of the last record in the file. Some programs fill all unused space in the last sector with this code. The EOF marker is not recognized by BDOS, but it acts as a signal to the transient not to read any further sectors, and to ignore the first EOF marker and all subsequent bytes in the buffer.

When random file access is in progress, the user program should always check the completion code returned by each Read or Write request, because BDOS distinguishes between an attempt to read a sector beyond the true end of file (code 1), and an attempt to read a sector which is within the file area but has no data in it (code 2). The latter condition could occur while building a tree, or while using hashed key techniques.

Deleting or Renaming a File

A user program can delete or rename a file by function calls to BDOS. The delete procedure is:

- Place the name and type of the file to be deleted in bytes 1 through 11 of an FCB (bytes are numbered from 0).
- Load Register C with the function code (19=13H=Delete).
- Load Register Pair DE with the address of the FCB.
- CALL BDOS.

No information is returned by BDOS after a Delete request.

The rename procedure is:

- Place the old name and type of the file to be renamed in bytes 1 through 11 of the FCB.
- Place the new name and file type in bytes 16 through 26 of the FCB.
- Load Register C with the function code (23=17H=Rename).
- Load Register Pair DE with the address of the FCB.
- CALL BDOS.

If BDOS finds a directory entry matching the filename and type in FCB bytes 1-11, it changes these to the filename and type specified in FCB bytes 16-26 and returns the byte address (within the sector) of the changed entry, in Register A. If no matching entry is found, BDOS returns 255 (0FFH) in Register A. The user program should check the completion code and take appropriate action if the renaming was not successful. After a successful renaming, a Close request must be issued for the file under the new name, otherwise the modified directory will not be written to the disk.

In the next issue of S-100 MICROSYSTEMS we will examine: "Implementing the IOBYTE Function".

S-100 MICROSYSTEMS' Bugs

The list of processor boards and manufacturers in the July/August issue was incomplete in that all three processor boards that IRISystems uses were omitted. These are:

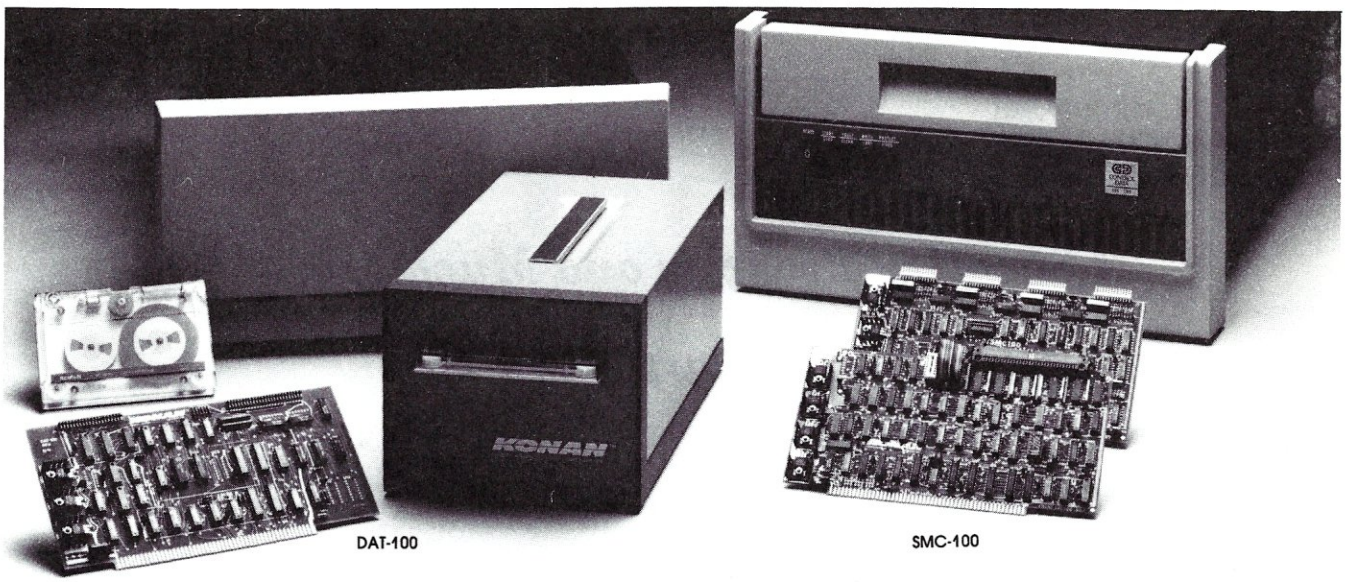
a. 8085 -TEI
5075 South Loop East
Houston TX 77033

b. Z80 -Industrial Micro Systems
628 North Eckhoff Street
Orange CA 92668

c. Z80 & M68000 -International Product Development Inc.
1708 Stierlin Road
Mountain View, CA 94043

Also, Digicomp Research's Pascal 100 contains both the Western Digital Microengine and a Z-80 processor.

Two portions of Chris Terry's CP/M Connection were transposed in the July/August issue. Corrections can be found on the first page of this month's installment "Part II-File Operations."



DAT-100

SMC-100

Hard disk and hardtape™ control

Up to 2400 Megabytes of hard disk control for the S-100 bus.

Konan's SMC-100 interfaces S-100 bus micro computers with all hard disk drives having the Industry Standard SMD Interface. It is available with software drivers for most popular operating systems. Each SMC-100 controls up to 4 drives ranging from 8 to 600 megabytes per drive, including most "Winchester" drives -- such as Kennedy, Control Data, Fujitsu, Calcomp, Microdata, Memorex, Ampex, and others.

SMC-100 is a sophisticated, reliable system for transferring data at fast 6 to 10 megahertz rates with onboard sector buffering, sector interleaving, and DMA.

SMC-100's low cost-per-megabyte advanced technology keeps your micro computer system micro-priced. Excellent quantity discounts are available.

Konan's HARDTAPE™ subsystem... very low cost tape and/or hard disk Winchester backup and more.

Konan's new DAT-100 Single Board Controller interfaces with a 17½ megabyte (unformatted) cartridge tape drive as well as the Marksman Winchester disk drive by Century Data.

The DAT-100 "hardtape" system is the only logical way to provide backup for "Winchester" type hard disk systems. (Yields complete hard disk backup with data verification in 20-25 minutes.)

Konan's HARDTAPE™ subsystem is available off the shelf as a complete tape and disk mass storage system or an inexpensive tape and/or disk subsystem.

Konan controllers and subsystems support most popular software packages including FAMOS™, CP/M® version 2.X, and MP/M.

Konan, first (and still the leader) in high-reliability tape and disk mass storage devices, offers OEM's, dealers and other users continuing diagnostic support and strong warranties. Usual delivery is off the shelf to 30 days with complete subsystems on hand for immediate delivery.

Call Konan's TOLL FREE ORDER LINE today:

800-528-4563

Or write to Bob L. Gramley
Konan Corporation, 1448 N. 27th Avenue
Phoenix, AZ 85009. TWX/TELEX 9109511552

CP/M® is a registered trade name of Digital Research,
FAMOS™ is a trade name of MVT Micro Computer Systems.
HARDTAPE™ is a trade name of Konan Corporation.

KONAN

NorthStar Topics

A Spooling Program and Much More

Part II

by Randy Reitz

In the last issue I described a program for a North Star disk system which implements input/output redirection, a printer SPOOL and a command stack. In order to redirect I/O to and from the disk as well as drive a printer with the contents of a disk file, this program must necessarily access disk files. I consider the part of the program which does North Star disk I/O to be a skeleton operating system. By my definition, an operating system manages the physical machine resources to provide an evironment to execute programs. One of the significant features of an operating system is the management of all input/output. The simpler an operating system makes I/O appear to the program environment, the easier it is to develop programs. One aspect of operating systems which distinguishes them is the specification of the program interface for input/output operations.

In the case of the North Star disk system, the I/O interface available for software development is extremely low level. The North Star DOS provides character I/O for terminals and other character oriented devices such as paper tape devices and line printers. No character I/O is available for disk files. The only access North Star DOS provides for disk files can be best described as the unbuffered do-it-yourself variety. North Star Basic does provide character by character as well as record access to disk files, but only for North Star Basic programs. For disk access, the North Star DOS provides a method for naming files and a method for accessing arbitrary blocks of data on the disk. North Star DOS does not provide any interface which will associate a file name with a particular area of the disk (open a file), read/write a byte or record and chose a file. The System Software Manual hints at how this can be done but no software is provided.

In order to implement input/output redirection and a printer spool, I needed to provide a means to buffer disk accesses. To understand this, you need to consider the type of input/output which the hardware controlling a disk device can accept. The most versatile type of disk controller will make the disk device appear as an extension of the computer's main memory. Hence, a limited amount of main memory can be made to appear to be much larger to an application program because the disk controller (and other hardware) is constantly swapping the machine's main memory with the memory available on the disk device. A simpler disk controller, such as the North Star, will only sort out the timing considerations necessary to locate a particular block of data (track and sector) on the rotating disk. Once the particular spot on the disk is found, all the information must be read or written as one block. The North Star DOS entry DCOM will find the requested spot on the disk and then read or write one or more consecutive blocks. The North Star disk controller is set up to recognize 10 such blocks on each track of a disk. The North Star DOS is set up to recognize 35 such tracks for a total of 350 blocks. In order to access one byte in a selected block, additional software is needed.

The process of accessing a single byte or record of data on a disk device requires a whole block of data to be buffered in the computer's main memory. In addition, the application program is greatly simplified if it can refer to a logical block in a named file and let the operating system sort out the actual physical block number, disk track and sector (another name for block) number. The second part of the spooling program implements a version of buffered disk I/O. I had never tried to implement this type of program before, and this experience proved to be very educational. I can now better appreciate the complexity of an operating system such as CP/M.

The CP/M operating system uses a 33-byte string of data to hold relevant information about an open disk file. This 33 byte string of data is called the File Control Block (FCB). The FCB is supplied by the application program and CP/M uses this memory as a scratch pad while it performs disk access. Most operating systems require some type of scratch pad when disk access is performed. I can remember doing battle with IBM's Job Control Language and specifying all that DCB (Data Control Block) information and never really understanding why it was needed. I kept thinking the computer should figure it all out. The only operating system I have used which does not require an area of the application program's memory for a disk I/O scratch pad is UNIX. The UNIX system is completely unique and I hope an affordable operating system of this type will be available someday for an S-100 type

Randy Reitz, 26 Maple St., Chatham Township, NJ 07928

computer. The operating system I am about to describe uses a 19-byte scratch pad. The layout is as follows:

ITEM	LENGTH	DESCRIPTION
XXFILE	12	Contains the name of the disk file
XXUNIT	1	Contains the disk drive number
XXPTR	1	Pointer to RAM buffer (low byte)
XXBUF	1	Pointer to RAM buffer (high byte)
XXADR	2	Disk address for next read/write
XXLEN	2	Disk file length
	19	

The longest filename allowed by North Star DOS is eight characters. I chose to include the drive specification in XXFILE, so this adds two more bytes (a comma and drive number). The filename is terminated with a CR (ASCII 13) and RUBOUT (I use FF hex) so a total of 12 bytes is required. When the disk drive number is stripped from the filename, it is stored in XXUNIT. When XXUNIT is zero, the file is considered closed. The pointer to the buffer used to temporarily hold the block of data is in the next two bytes. The data blocks on North Star disks are 256 bytes in single density. This is a very convenient number since exactly one byte is needed to indicate the next position in the buffer to be read/written. This convenience disappears with double density (which is why the program won't work with double density). To compensate I used this trick in several places:when the low byte overflows-it's time to get/put another block. Now that I reflect on this decision, using tricks such as this (sometimes called magic numbers) makes a program depend too much on the particular hardware that it runs on. Finally, the scratch pad contains two numbers relevant to the particular disk file. The first contains the address of the next physical block which will be read/written when the buffer overflows. The second number is initially the total length (in blocks) of the disk file. The disk address can range up to 349 and the length can range from 0 to 345 (the first 4 blocks on every North Star disk contain the directory), so two bytes are needed for each number.

With these preliminaries out of the way, we can discuss how the program works. The character-by-character write routine starts at location WRDISK. Throughout the program, access to the information in the scratch pad is made through addresses contained in the 8080 machine registers. This allows the scratchpad and buffer to be located (and relocated) anywhere in memory. Each routine starts with comments indicating the information expected in the registers. Error conditions are indicated using the 8080 flag register. Usually the carry bit set indicates an error and the state of the zero flag and minus/plus flag will indicate the type of error. The WRDISK simply saves the byte to be written (B-register) in the buffer and increments the pointer (low byte of the buffer address). If the buffer is not full, the routine returns. If the buffer is full, the routine sets up the registers for the DODSK routine which writes the buffer to disk. Any error in DODSK will be passed back to the calling program by WRDISK.

The RDDISK is very similar to WRDISK and will return the next character from the buffer in the A-register. If all the characters in the buffer have been read, the registers are set up and DODSK is called to fill the buffer. The DODSK routine does the unbuffered disk I/O. This routine uses the DCOM entry in the North Star DOS after the information contained in the scratch pad is updated and all the 8080 registers are set up. The DODSK routine also checks for a length error (trying to read/write more blocks than the file contains). The file length byte is decremented for each block read/written. When this byte in the scratch pad is zero, a length error is indicated. If the North Star DOS routine DCOM indicates an error, DODSK interprets the error to be that the disk is write-protected. This is a good guess if you are sure the information in the 8080 registers upon entry to DCOM is correct. Unfortunately, the System Software Manual does not tell you this.

The SEARCH and CREATE routines are the ones necessary to open a disk file. I don't have an open routine in the operating system. The disk file open is done in the spool driver routines discussed in the last issue. I developed an open routine subsequent to writing this program and I have used it for other work. To open a file, the following steps are needed:1) set up a 19-byte scratch pad and insert the filename, 2) call SEARCH to see if the file exists on the selected disk device (extracted from filename), and 3) call CREATE if the file does not exist. If SEARCH finds the file, the starting disk address and total file length are loaded into the scratch pad. If SEARCH is unsuccessful, the next available address on the disk is in the DE-register pair. This is where CREATE expects to find it so SEARCH must be called before CREATE. Also the CREATE routine requires the desired file type to be in the A-register and the desired file length (in blocks) to be in the BC-register pair. If CREATE is called with a requested length of zero (BC = O) then all available space on the disk will be used. When this file is closed, the length will be adjusted to the number of blocks actually used. This is the closest I could come to dynamic files in a North Star system. The CREATE routine will determine if the disk is too full to contain the requested file, if the file name is bad and if the disk is write-protected.

The last routine in the operating system is CLSIT which closes a disk file. This routine will put an end-of-file byte (ASCII 1, SOH) in the buffer and write the final buffer to disk. The length of the file contained in the disk directory will be adjusted if the dynamic file sizing was requested. The XXUNIT byte in the scratch pad is zeroed to indicate that the file is closed. If the file was open for reading only, the CLSIT routine should not be used since this file can be closed merely by zeroing the XXUNIT byte. The CLSIT routine will signal an error if the disk in the drive has been disturbed since opening the file.

The remainder of the code in the program is service routines for the operating system and disk spool parts of the program. The largest block of code handles errors when the operating system detects disk problems. Smart error routines are difficult to write. These error routines attempt to give a good message and return after cleaning things up as much as possible. The program ends with a data section which allocates space for program variable storage, the three 19-byte scratch pads for the write, read and output

NEWS & VIEWS

by Sol Libes

MICROSOFT SIGNS UNIX AGREEMENT

Microsoft, Bellevue Wash., has signed an agreement with Western Electric for the rights to develop and market a version of UNIX, developed by Bell Labs. Their version will be specifically for 16-bit microprocessors, such as the 8086, Z8000 and 68000 and will be called "XENIX". UNIX is probably "the" most popular minicomputer time-sharing operating system in surrent use. It is very popular in the educational community, most probably because WE sold it to these institions for a very low fee. However, it has been gaining in popularity in the business world as well.

The charge for the software package will be based on volume and number of users. The initial fee for a four user system will range from $500 to $30000. Release of the package is expected in early 1981.

MICROSOFT RELEASES BASIC FOR Z8000

Microsoft again strikes a first with its implementation of Basic for Z8000 based systems. This is the first worthwhile software package to be released for the Z8000. Microsoft was able to gain this big lead by developing the package on a large minicomputer system using a cross-assembler and several development programs. Microsoft's timing could not have been better as the first Z8000 CPU cards and systems are only just now becoming available.

The Basic will have the same features as Microsoft's Basic for 8-bit machines and will execute about five times faster. Microsoft's marketing emphasis for the Z8000 Basic will be primarily to OEMs.

RUMORS

Digital Equipment Corp. (DEC) will soon release a 16-bit microprocessor chip that will be compatible with 8080/Z80/6800 support ICs. It will have the power of a PDP-11/23. At least one outfit is presently investigating an S-100 implementation with plans of running the popular RT-11 time-sharing operating system.

CPM USER GROUP LIBRARY NOW ON NORTH STARDISK

Finally, the largest and best microcomputer software library has been made available on North Star Disk. The CP/M Users Group, a non-profit opertion, is making available all 42 volumes of the CP/M Software Library. Until now, it was only available on 8" disks. These disks may be copied without restriction. The usual practice is for a group or club to bus a set of disks, then each member makes a copy from this master set.

The disks are available in either double density Version 1.4 or 2.2 and guad density Version 2.2 formats. They are not available in single density formats. A catalog of the volumes is available for $6 ($11 overseas). The disks are either $8 or $12 per volume depending on whether they require one or two disks per volume (overseas $12 or $16). For information and ordering contact: CP/M Users Group, 1651 Third Ave., New York, NY 10028.

```
SYMBOL TABLE

BADCL 1BF0    BADES 1C18    BADNM 1BCC    BADPA 1B86    BADTY 1BBE
BELL  0007    BLANK 1C10    BS    0008    CC    00EB    CIN   2010
CKEYB 17A9    CKMOT 1B22    CLOSE 1BDC    CLSER 1B5D    CLSIT 1A8A
CLSPO 1852    COUT  200D    CR    000D    CREAT 19F3    CRLF  1BED
CTRLC 0003    CTRLS 0013    CURR  1901    DCOM  2022    DELET 1773
DIRLO 1AD7    DISPL 194F    DLOOK 201C    DOCRT 170F    DODSK 19A8
DOPRT 170C    DOSEN 2028    DOSER 202C    DOSPO 160B    DOSPT 28B7
DRVSE 2003    DSKFL 1A71    DSKFU 1A72    DSKNA 1A79    DSKWP 1A80
DSPBE 194D    DWRIT 201F    ESC   001B    ESCER 1B79    FETCH 165F
FILNM 1B09    FIN   1659    FLERR 1B4A    FULL  1BB4    GETCH 1936
GETCP 17F5    GETIT 1996    GETSP 1786    GETST 1948    GNAME 1880
GOSTK 172B    IODEV 1C55    IOFLA 1C54    IOSTK 1CFF    KBDMS 1604
KEYBD 168C    KILL  173C    KILLS 1780    LDSTK 1744    LF    000A
LNERR 1B3D    LOAD  174A    LOADH 192D    LOCST 1C95    LOOP  1652
MO    0010    MOVDI 19E7    MOVLN 1A2C    MOVNM 1A41    NAME  1898
NEW   1913    NF    1BA7    NMERR 1B51    NOSEL 0059    NPERR 1B50
OK1   17E4    OK2   17F3    OKLEN 19C3    OKNAM 18AD    OUTSP 1716
PMSG  1C2E    PRACT 1807    PRFIL 1C82    PRINT 18F5    PRLOO 1818
PRNAM 187B    PRUNI 1C8E    PUTCR 1958    PUTSP 17B7    RDDIS 197F
RDERR 1B43    RDFIL 1C6F    RDNAM 186C    RDNF  1B9D    RDSTK 1C50
RDUNI 1C7B    READA EB10    READN 18DC    RESET EB90    RETM  18D3
RUBOU 007F    SAVER 1C56    SAVPT 173F    SAVST 1C58    SEARC 19D3
SELEC 1B2C    SKIP  16DB    SPLBU 1D00    SPLCI 1644    SPLCO 16DF
SPLFI 1C5C    SPLPR 1600    SPLSR 1844    SRCON 19DF    STACK 1C52
STATP 1603    STOP  1C3C    STOPR 1C39    STRTM 1B16    TESTS 1655
TOLON 1BFE    TYERR 1B57    UPDAT 1926    USEBC 1A1D    USRCI 1608
USRCO 1605    WRDIR 1A5B    WRDIS 195D    WRERR 1B37    WRFIL 1C5C
WRITE 18E8    WRNAM 1874
```

North Star cont'd...

spool files, the command stack space and finally the 8080 machine stack.

As I pointed out at the start of this article, this program was my first attempt to handle disk input/output. As I reflect on this program I probably would have done many things differently. Nevertheless, I hope you can use this as an example of some of the considerations required when you do disk I/O with the North Star DOS. Maybe this exercise will convince you to buy CP/M for your North Star system. This program will only give you I/O redirection, etc. with North Star DOS. If you want these features and don't want to type this program in, I can supply you a copy on disk for $15. I will include the source code and the assembled version shown here. □

North Star Horizon–
COMPUTER WITH CLASS

The North Star Horizon computer can be found everywhere computers are used: business, engineering, home — even the classroom. Low cost, performance, reliability and software availability are the obvious reasons for Horizon's popularity. But, when a college bookstore orders our BASIC manuals, we know we have done the job from A to Z.

Don't take our word for it. Read what these instructors have to say about the North Star Horizon:

"We bought a Horizon not only for its reliability record, but also because the North Star diskette format is the industry standard for software exchange. The Horizon is the first computer we have bought that came on-line as soon as we plugged it in, and it has been running ever since!"

— Melvin Davidson, Western Washington University, Bellingham, Washington

"After I gave a ½ hour demonstration of the Horizon to our students, the sign-ups for next term's class in BASIC jumped from 18 to 72."

— Harold Nay, Pleasant Hill HS, Pleasant Hill, California

"With our Horizon we brought 130 kids from knowing nothing about computers to the point of writing their own Pascal programs. I also use it to keep track of over 900 student files, including a weekly updated report card and attendance figures."

— Armando Picciotto, Kennedy HS, Richmond, California

"The Horizon is the best computer I could find for my class. It has an almost unlimited amount of software to choose from. And the dual diskette drives mean that we don't have to waste valuable classroom time loading programs, as with computers using cassette drives."

— Gary Montante, Ygnacio Valley HS, Walnut Creek, Calif.

See the Horizon at your local North Star dealer.

NorthStar

North Star Computers, Inc.
1440 Fourth Street
Berkeley, CA 94710
(415) 527-6950
TWX/Telex 910-366-7001

A Program for Renaming Files on North Star Disks

by Mark M. Zeiger

Presently, to change the name of a file on a North Star formatted disk using the standard DOS commands, these instructions are needed:

1. CReate (new file name) (length)
2. TYpe (new file name) (type) (address)
3. CF (old file) (new file)
4. DElete (old file)

and probably

5. COmpact (if you have release 4 or 5 DOS, you will have to load COMPACT as a separate program).

The program described in this article will rename a file without changing anything on the disk except the directory entry for the file being renamed. Directions are given at the beginning of the program and the user is prompted throughout the program's run. Anytime the program is waiting for a filename, typing a Control-D will list the directory (and restart the program), a Control-X will restart the program, and a Control-C will take the user back to North Star DOS.

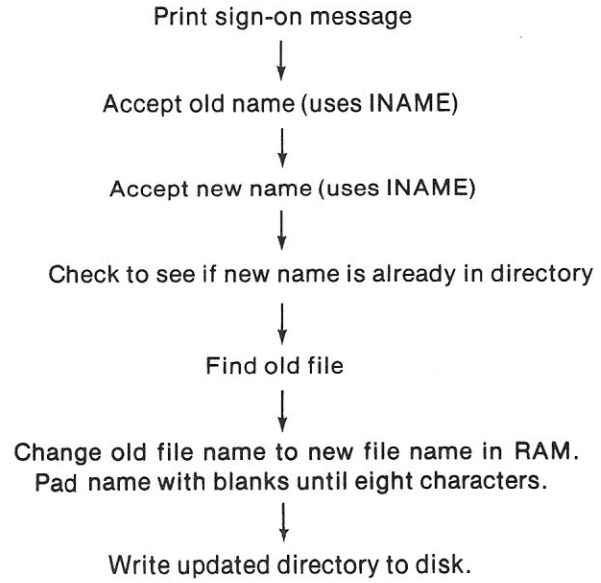
The program first asks for the drive number. The user has the option of typing in the drive number (the 'return' key is not needed) or typing Control-C to return to DOS. If a drive is selected, the program to be renamed is then entered followed by a return. If the name is over eight characters, an error occurs and the prompt for the file to be renamed is given again. The program then asks for the new name of the file, again accepting a name of at least one but not more than eight characters. "Delete" (7F hex) or "Rubout" (5FH) may be used to erase a character when inputting names. The program will automatically convert lowercase characters to uppercase. There is a built-in safety check that will not allow entry of just a "Return" when asked for the new name since to do so would be tantamount to erasing the file from the directory (as I unfortunately discovered). After the new name is entered, the file will be updated and the program will restart itself.

Mark M. Zeiger, 198-01B 67th Ave., Flushing, NY 11365

If the new name already exists on the disk, the user will be informed of the fact and be asked to pick another name. If the file being renamed is not on the disk, an appropriate message will be given, the directory will automatically be listed, and the user will have to type in both a correct old name and a new name. If there is any doubt as to a name, type Control-D to list the directory.

The RENAME program uses the machine language routines DLOOK, DWRIT, and DLIST as described in the DOS section of the North Star manual. The source program is annotated for clarity (hopefully). Since this is one of my earlier efforts at machine language programming, the code is not as structured as it could be, but since it works perfectly I am not going to rewrite it.

The program flow is as follows:



The major subroutine is INAME which calls input, places the name in the correct buffers (old or new), performs checks on the names, allows deletion of characters, and accepts Control D, X, or C.

You will have to change two lines in the source code to adapt the program to your system. Line 0100 should signify your console backspace character and 0110 should contain the number of drives on your system. If you have a non-standard DOS, you will have to change the equates at the end of the program. If your system has only one drive, you may delete the section which asks for the drive number. Delete lines 150 to 290, change the label in line 300 to "START", and change line 340 to "JC START".

The routines in North Star DOS make this program very easy to implement. The simplest routine to use is the LIST routine that lists the disk directory. This routine is called with the drive number in the A-register and the output device number in the L-register (the latter is not used in RENAME). The DOS routines which really do the work of this program are DLOOK and DWRIT. To use DLOOK the Accumulator must again contain the drive number and the HL-registers should point to the RAM address containing the first letter of the file name being sought. The name in RAM should have either a blank or return at its end. When DLOOK is called, the entire directory is written into RAM in the DOS area. If the name is found, the carry flag is reset and the HL register pair contains the address of the last (8th) character of the file name in the DOS directory area of RAM. If the name does not exist then the HL register pair contains the first free disk address (this would be used when creating a new file). Any of the directory entries in RAM may now be changed (although only names, types, and GO addresses may be changed without destroying the integrity of the directory). The DOS command DWRIT may then be used to write the changed directory in RAM back onto the disk. No disk activity should take place between using DLOOK and DWRIT and no parameters are needed when using DWRIT.

RENAME uses the above routines in the following manner. Lines 320, 330, 370, and 380 load the old and new file name into buffers called OBUF and NBUF respectively. Lines 430 through 490 then use DLOOK to make sure the new name is not already being used. In this case the program will continue if DLOOK returns with the carry flag set. Lines 530 to 560 use DLOOK to determine if the old name is on the disk. Hopefully the routine will return with the carry flag not set. If the old name is on the disk, then the HL register pair must be decreased by eight (lines 690 and 700) since the register pair is pointing to the end of the name. Lines 740 through 920 transfer the name in NBUF to the DOS directory in RAM and lines 960 and 970 use DWRIT to put the altered directory back on the disk.

Note that the hex code listing stops at 2E60. The code from this point on are the ASCII messages. The assembler prints these on multiple lines making the source code difficult to read. Therefore, only the source code was printed out from this point on. □

```
2D00                    0010 ; RENAME PROGRAM     BY MARK M. ZEIGER
2D00                    0020 ; RENAMES FILE ON NORTH STAR FORMATTED DISK
2D00                    0030 ;THIS PROGRAM WORKS FOR BOTH SINGLE AND DOUBLE DENSITY DOS.
2D00                    0040 ;ASSEMBLE AT 2A00H FOR SINGLE DENSITY AND 2D00H FOR DOUBLE DENSITY.
2D00                    0050 ;
2D00                    0060 ; ----------> IMPORTANT <----------
2D00                    0070 ;
2D00                    0080 ;  CHANGE THE NEXT TWO LINES TO SUIT INDIVIDUAL SYSTEM.
2D00                    0090 ;
2D00                    0100 BS      EQU   8              ;TERMINAL BACKSPACE CHARACTER
2D00                    0110 DRNUM   EQU   2              ;NUMBER OF DRIVES ON SYSTEM
2D00                    0120 ;
2D00 21 EA 2E           0130 BEGIN   LXI   H,TITLE        ;PRINT SIGN-ON MESSAGE
2D03 CD A3 2D           0140         CALL  MESG           ;MESSAGE OUTPUT ROUTINE.
2D06 21 C2 2F           0150 START   LXI   H,MESG0        ;ASKS WHICH DRIVE FILE IS ON
2D09 CD A3 2D           0160         CALL  MESG
2D0C CD 10 20           0170         CALL  INPUT
2D0F FE 03              0180         CPI   03H            ;RETURN TO DOS IF CTRL-C.
2D11 CA 2E 2E           0190         JZ    GO             ;OUTPUTS CR LF AND JUMPS TO DOS.
2D14 F5                 0200         PUSH  PSW
2D15 47                 0210         MOV   B,A
2D16 CD 19 2E           0220         CALL  OUTP
2D19 F1                 0230         POP   PSW
2D1A DC 30              0240         SUI   30H            ;CHANGE DRIVE NUMBER TO BINARY
2D1C FE 01              0250         CPI   1              ;NEXT FOUR INSTRUCTIONS..
2D1E DA 06 2D           0260         JC    START          ;..CHECK FOR CORRECT DRIVE NUMBER.
2D21 FE 03              0270         CPI   DRNUM+1
2D23 D2 06 2D           0280         JNC   START
2D26 32 58 2E           0290         STA   DRIVE
2D29 21 59 2E           0300 OLD     LXI   H,MESG1        ;ASKS FOR FILES BEING RENAMED.
2D2C CD A3 2D           0310         CALL  MESG
2D2F 21 46 2E           0320         LXI   H,OBUF         ;ADDRESSES OLD FILE NAME BUFFER
2D32 CD AF 2D           0330         CALL  INAME          ;ROUTINE PUTS NAME IN CORRECT BUFFER
2D35 DA 29 2D           0340         JC    OLD            ;IF CARRY SET, NAME IS TOO LONG.
2D38 21 76 2E           0350 NEW     LXI   H,MESG2        ;ASKS FOR NEW NAME
2D3B CD A3 2D           0360         CALL  MESG
2D3E 21 4F 2E           0370         LXI   H,NBUF         ;ADDRESSES NEW FILE NAME BUFFER
2D41 CD AF 2D           0380         CALL  INAME
2D44 DA 38 2D           0390         JC    NEW            ;IF CARRY SET, NAME WAS TOO LONG
2D47                    0400 ;
2D47                    0410 ; CHECK TO SEE IF NEW NAME ALREADY EXISTS - USES DLOOK
2D47                    0420 ;
2D47 3A 58 2E           0430         LDA   DRIVE
2D4A 21 4F 2E           0440         LXI   H,NBUF
2D4D CD 1C 20           0450         CALL  DLOOK
2D50 DA 5C 2D           0460         JC    LOKUP          ;IF CARRY SET, NAME WAS NOT FOUND
2D53 21 B9 2E           0470         LXI   H,MESG4        ;MESSAGE THAT NAME ALREADY EXISTS
2D56 CD A3 2D           0480         CALL  MESG
2D59 C3 38 2D           0490         JMP   NEW            ;GET ANOTHER NEW NAME
```

Renaming Files cont'd...

```
2D5C                    0500 ;
2D5C                    0510 ; FIND OLD NAME IN DIRECTORY
2D5C                    0520 ;
2D5C 3A 58 2E           0530 LOKUP  LDA   DRIVE         ;PUT DISK UNIT NUMBER IN ACCUMULATOR
2D5F 21 46 2E           0540        LXI   H,OBUF        ;HL MUST POINT TO FILE NAME IN RAM
2D62 CD 1C 20           0550        CALL  DLOOK
2D65 D2 77 2D           0560        JNC   FOUND         ;CARRY IS SET IF FILE IS NOT FOUND
2D68 21 A5 2E           0570        LXI   H,MESG3       ;MESSAGE THAT FILE WAS NOT FOUND


2D6B CD A3 2D           0580        CALL  MESG
2D6E                    0590 ;AUTOMATICALLY LIST DIRECTORY IF NAME NOT FOUND.
2D6E                    0600 ;DELETE NEXT TWO INSTRUCTIONS IF FEATURE NOT DESIRED.
2D6E 3A 58 2E           0610        LDA   DRIVE         ;LOAD DISK UNIT NUMBER
2D71 CD 25 20           0620        CALL  DLIST         ;LIST THE DIRECTORY
2D74                    0630 ;
2D74 C3 06 2D           0640        JMP   START
2D77                    0650 ;
2D77                    0660 ;DECREASE HL BY 8 SO THAT IT POINTS TO BEGINNING OF
2D77                    0670 ;FILE ENTRY IN DOS RAM
2D77                    0680 ;
2D77 11 F8 FF           0690 FOUND  LXI   D,-8
2D7A 19                 0700        DAD   D
2D7B                    0710 ;
2D7B                    0720 ; MOVE NEW FILE NAME FROM NBUF TO DOS RAM ADDRESSED BY HL-REG
2D7B                    0730 ;
2D7B 0E 09              0740        MVI   C,9
2D7D 11 4F 2E           0750        LXI   D,NBUF        ;POINT DE TO BEGINNING OF NEW NAME BUFFER
2D80 1A                 0760 TRANS  LDAX  D             ;MOVE NEW NAME CHARACTER TO ACC
2D81 0D                 0770        DCR   C
2D82 FE 0D              0780        CPI   0DH           ;LAST CHARACTER?
2D84 CA 90 2D           0790        JZ    SPACE
2D87 77                 0800        MOV   M,A           ;MOVE NEW NAME TO DOS RAM
2D88 23                 0810        INX   H
2D89 13                 0820        INX   D
2D8A 3E 00              0830        MVI   A,0
2D8C B9                 0840        CMP   C
2D8D C2 80 2D           0850        JNZ   TRANS
2D90 3E 00              0860 SPACE  MVI   A,0           ;COMPARE C-REG TO ZERO
2D92 B9                 0870        CMP   C             ;HAVE EIGHT CHARACTERS BEEN MOVED?
2D93 CA 9D 2D           0880        JZ    NOSPC
2D96 36 20              0890        MVI   M,' '         ;FILL DOS RAM WITH BLANKS UNTIL 8 CHARACTERS
2D98 23                 0900        INX   H
2D99 0D                 0910        DCR   C
2D9A C3 90 2D           0920        JMP   SPACE
2D9D                    0930 ;
2D9D                    0940 ;UPDATE THE DIRECTORY
2D9D                    0950 ;
2D9D CD 1F 20           0960 NOSPC  CALL  DWRIT         ;NORTH STAR ROUTINE TO UPDATE DIRECTORY
2DA0 C3 06 2D           0970        JMP   START
2DA3                    0980 ;
2DA3                    0990 ; -------> SUBROUTINES <-------
2DA3                    1000 ;
2DA3 46                 1010 MESG   MOV   B,M           ;NORTH STAR DOS WANTS OUTPUT IN B-REG
2DA4 3E 00              1020        MVI   A,0
2DA6 CD 19 2E           1030        CALL  OUTP
2DA9 B8                 1040        CMP   B             ;TEST FOR END OF MESSAGE
2DAA C8                 1050        RZ
2DAB 23                 1060        INX   H
2DAC C3 A3 2D           1070        JMP   MESG
2DAF                    1080 ;
2DAF                    1090 ;SUBROUTINE  ----> INAME
2DAF                    1100 ;
2DAF                    1110 ;PUTS OLD AND NEW FILE NAMES INTO RESPECTIVE BUFFERS.
2DAF                    1120 ;PERFORMS VARIOUS TESTS TO MAKE SURE THAT:
2DAF                    1130 ;   NAME IS NOT LONGER THAN EIGHT CHARACTERS
2DAF                    1140 ;   NAME HAS AT LEAST ONE CHARACTER


2DAF                    1150 ;   LOWER CASE IS CHANGED TO UPPER CASE
2DAF                    1160 ;   SPACES OR CONROL CHARACTERS ARE NOT PART OF NAME
2DAF                    1170 ;ALLOWS DELETION OF CHARACTERS
2DAF                    1180 ;ALLOWS JUMPS TO LIST DIRECTORY, GO TO N.S. DOS, OR RESTART OF PROGRAM.
2DAF                    1190 ;
2DAF 0E 0A              1200 INAME  MVI   C,10          ;COUNTS OUT 8 CHARACTERS AND CR FOR NAMES
2DB1 CD 10 20           1210        CALL  INPUT
2DB4 FE 03              1220        CPI   3             ;CTRL-C RETURNS CONTROL TO N.S. DOS
2DB6 CA 1F 2E           1230        JZ    LEAVE
2DB9 FE 18              1240        CPI   18H           ;CTRL-X RESTARTS PROGRAM
2DBB CA 1F 2E           1250        JZ    LEAVE
2DBE FE 04              1260        CPI   4             ;CTRL-D LISTS DIRECTORY
2DC0 CA 1F 2E           1270        JZ    LEAVE
2DC3 FE 0D              1280        CPI   0DH           ;IS CHARACTER A CARRIAGE RETURN?
2DC5 CA CD 2D           1290        JZ    SKIP1         ;SKIP NEXT TWO INSTRUCTIONS IF IT IS.
2DC8 FE 21              1300        CPI   21H           ;IS CHARACTER A SPACE OR LESS?
2DCA DA B1 2D           1310        JC    INAME+2       ;DON'T ACCEPT IF IT IS.
2DCD FE 60              1320 SKIP1  CPI   60H           ;IS CHARACTER LOWER CASE?
2DCF DA D4 2D           1330        JC    SKIP2         ;SKIP NEXT INSTRUCTION IF IT IS.
```

Renaming Files cont'd...

```
2DD2 E6 DF              1340          ANI   0DFH          ;MAKE LOWER CASE UPPER CASE
2DD4 FE 5F              1350 SKIP2    CPI   RUBOT         ;WAS IT A RUBOUT?
2DD6 CA 05 2E           1360          JZ    DELET         ;'ANI 0DFH' MAKES 'DELETE (7FH)' A RUBOUT.
2DD9                    1370 ;
2DD9                    1380 ;THE NEXT 7 INSTRUCTIONS MAKE SURE CARRIAGE RETURN IS NOT
2DD9                    1390 ;THE FIRST CHARACTER OF NEW NAME. THAT WOULD DESTROY THE FILE.
2DD9                    1400 ;
2DD9 FE 0D              1410          CPI   0DH           ;CARRIAGE RETURN?
2DDB C2 E6 2D           1420          JNZ   SKIP3         ;SKIP NEXT FIVE INSTRUCTIONS IF NOT
2DDE 57                 1430          MOV   D,A           ;SAVE THE ACC
2DDF 3E 0A              1440          MVI   A,10
2DE1 B9                 1450          CMP   C
2DE2 CA AF 2D           1460          JZ    INAME         ;CR WAS FIRST CHARACTER - DO OVER
2DE5 7A                 1470          MOV   A,D           ;RESTORE THE ACC
2DE6                    1480 ;
2DE6 77                 1490 SKIP3    MOV   M,A           ;IT'S A GOOD CHARACTER. MOVE IT TO BUFFER.
2DE7 47                 1500          MOV   B,A           ;OUTPUT IT
2DE8 CD 19 2E           1510          CALL  OUTP
2DEB 0D                 1520          DCR   C
2DEC CA FD 2D           1530          JZ    ERROR         ;NAMES MAY ONLY BE 8 CHARACTERS LONG + CR
2DEF 23                 1540          INX   H
2DF0 FE 0D              1550          CPI   0DH           ;FINISHED ENTERING NAME?
2DF2 C2 B1 2D           1560          JNZ   INAME+2
2DF5 21 A2 2E           1570          LXI   H,CRLF
2DF8 CD A3 2D           1580          CALL  MESG
2DFB B7                 1590          ORA   A             ;CLEAR CARRY - NO ERROR
2DFC C9                 1600          RET
2DFD                    1610 ;
2DFD 21 8F 2E           1620 ERROR    LXI   H,ERMSG       ;MESSAGE SAYS TOO MANY CHARACTERS
2E00 CD A3 2D           1630          CALL  MESG
2E03 37                 1640          STC                 ;CARRY SET MEANS ERROR
2E04 C9                 1650          RET
2E05                    1660 ;
2E05 0C                 1670 DELET    INR   C             ;INCREASE COUNT SINCE NAME IS SHORTENED
2E06 3E 0B              1680          MVI   A,11          ;DON'T DELETE PAST START OF NAME
2E08 B9                 1690          CMP   C
2E09 CA 15 2E           1700          JZ    NODEL
2E0C 2B                 1710          DCX   H             ;MOVE BUFFER POINTER BACK ONE SPACE
2E0D 06 08              1720          MVI   B,BS          ;OUTPUT TERMINAL BACKSPACE CHARACTER
2E0F CD 19 2E           1730          CALL  OUTP
2E12 C3 B1 2D           1740          JMP   INAME+2
2E15 0D                 1750 NODEL    DCR   C             ;FORGET ABOUT DELETE
2E16 C3 B1 2D           1760          JMP   INAME+2
2E19                    1770 ;
2E19 F5                 1780 OUTP     PUSH  PSW
2E1A CD 0D 20           1790          CALL  COUT
2E1D F1                 1800          POP   PSW
2E1E C9                 1810          RET
2E1F                    1820 ;
2E1F                    1830 ;"LEAVE" IS ONLY ENTERED IF CTRL C, X, OR D IS TYPED.
2E1F                    1840 ;
2E1F 33                 1850 LEAVE    INX   SP            ;NORMALIZE STACK SINCE A SUBROUTINE
2E20 33                 1860          INX   SP            ;WAS LEFT WITHOUT A RETURN.
2E21 FE 03              1870          CPI   3             ;OUTPUT CR/LF, THEN RETURN TO DOS.
2E23 CA 2E 2E           1880          JZ    GO
2E26 FE 04              1890          CPI   4             ;IF CTRL-D, LIST DIRECTORY
2E28 CA 37 2E           1900          JZ    DIR
2E2B C3 06 2D           1910          JMP   START         ;IT'S A CTRL-X. RESTART PROGRAM.
2E2E 21 A2 2E           1920 GO       LXI   H,CRLF
2E31 CD A3 2D           1930          CALL  MESG
2E34 C3 28 20           1940          JMP   DOS           ; GO BACK TO NORTH STAR DOS
2E37 21 A2 2E           1950 DIR      LXI   H,CRLF
2E3A CD A3 2D           1960          CALL  MESG
2E3D 3A 58 2E           1970          LDA   DRIVE
2E40 CD 25 20           1980          CALL  DLIST         ;LIST DIRECTORY
2E43 C3 06 2D           1990          JMP   START
2E46                    2000 ;
2E46                    2010 ; EQUATES
2E46                    2020 ;
2E46                    2030 OBUF     DS    9             ;BUFFER FOR OLD NAME. 8 CHARACTERS + CR
2E4F                    2040 NBUF     DS    9             ;BUFFER FOR NEW NAME
2E58 01                 2050 DRIVE    DB    1
2E59 0A 0D              2060 MESG1    DW    0D0AH
                        2070          ASC   'ENTER FILE TO BE RENAMED: '
                        2080 ;
                        2090          DB    0             ;END OF MESSAGE INDICATOR
                        2100 ;
                        2110 MESG2    ASC   'ENTER NEW NAME OF FILE: '
                        2120          DB    0
                        2130 ERMSG    DW    0D0AH
                        2140          ASC   'NAME IS TOO LONG.'
                        2150 CRLF     DW    0D0AH
                        2160          DB    0
                        2170 MESG3    DW    0D0AH
                        2180          ASC   'FILE NOT FOUND'
                        2190          DW    0D0AH
                        2200          DW    000AH
                        2210 MESG4    DW    0D0AH
                        2220          ASC   'FILE NAME ALREADY EXISTS - PICK ANOTHER NAME'
                        2230          DW    0D0AH
                        2240          DB    0
                        2250 TITLE    DW    0D0AH
                        2260          ASC   'THIS PROGRAM RENAMES FILES ON NORTH STAR FORMATTED DISKS.'
```

Renaming Files cont'd...

```
2270            DW    0D0AH
2280            ASC   'CONTROL-C EXITS TO DOS     CONTROL-X RESTARTS PROGRAM'
2290            DW    0D0AH
2300            ASC   '              CONTROL-D LISTS DIRECTORY'
2310            DW    0D0AH
2320            ASC   '  RUBOUT OR DELETE WILL DELETE THE LAST CHARACTER TYPED'
2330            DW    0A0AH
2340            DW    0D0AH
2350            DB    0
2360 MESG0      DW    0D0AH
2370            ASC   'ENTER DRIVE NUMBER (CTRL-C TO RETURN TO DOS): '
2380            DB    0
2390 COUT       EQU   200DH           ;NORTH STAR COUT JMP
2400 INPUT      EQU   2010H           ;NORTH STAR CIN JMP
2410 DLOOK      EQU   201CH
2420 DWRIT      EQU   201FH
2430 DLIST      EQU   2025H
2440 DOS        EQU   2028H
2450 RUBOT      EQU   5FH
2460 STOP       END
```

SYMBOL TABLE

```
BEGIN 2D00    BS    0008    COUT  200D    CRLF  2EA2    DELET 2E05
DIR   2E37    DLIST 2025    DLOOK 201C    DOS   2028    DRIVE 2E58
DRNUM 0002    DWRIT 201F    ERMSG 2E8F    ERROR 2DFD    FOUND 2D77
GO    2E2E    INAME 2DAF    INPUT 2010    LEAVE 2E1F    LOKUP 2D5C
MESG  2DA3    MESG0 2FC2    MESG1 2E59    MESG2 2E76    MESG3 2EA5
MESG4 2EB9    NBUF  2E4F    NEW   2D38    NODEL 2E15    NOSPC 2D9D
OBUF  2E46    OLD   2D29    OUTP  2E19    RUBOT 005F    SKIP1 2DCD
SKIP2 2DD4    SKIP3 2DE6    SPACE 2D90    START 2D06    STOP  2FF3
TITLE 2EEA    TRANS 2D80
```


Software Consultants Harken!

Do you do software customizing? Do you do software package installation? Do you provide turnkey systems? Do you run a programming shop? Full-time? Part-time?

Starting with a future issue **MICRO-SYSTEMS** will publish a directory of "software shops." Places where computer customers can go for software package customizing. Most purchases of computer software packages find that the package does not exactly meet their needs—customizing is required—sometimes this is minor and sometimes it is major. Our directory will provide low cost advertising for "software shops." If you wish to be listed write to MICRO-SYSTEMS, Box 1192, Mountainside, NJ 07902 for information.

S-100 MICROSYSTEMS Book Service

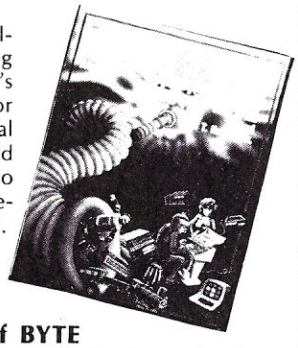
An Introduction to Microcomputers, Vol 0 - The Beginners Book

Adam Osborne. Parts of a computer and a complete system; binary, octal and hexadecimal number systems; computer logic; addressing and other terminology are discussed in a language the absolute beginner can understand. Hundreds of illustrations and photographs. 220 pp. $7.95 [9T].

Microcomputer Design

Donald P. Martin. This book is well-suited for the engineer who's designing microcomputers into his company's products. Not just block diagrams or vague theory, but dozens of practical circuits with schematics for CPUs based on 8008 chips, Includes interfacing to A/D, D/A, LED digits, UARTs, teletypewriters. Over 400 pp. ~~$14.95~~ [9P].

NOW $11.95



An Introduction to Microcomputers, Vol 1 - Basic Concepts

Adam Osborne. Thoroughly explains hardware and programming concepts common to all micro-processors: memory organization, instruction execution, interrupts, I/O, instruction sets and assembly programming. One of the best selling computer texts worldwide. 350 pp. $9.50 [9K].

Vol 2 - Some Real Microprocessors. Vol 3 - Real Support Devices

Adam Osborne. These volumes complement Volume 1. Vol. 2 discusses the operation of each of the following MPUS in detail: F8, SC/MP, 8080A, Z80, 6800, PPS-8, 2650, COS MAC, 9002, 6100 and seven others. Also information on selecting a micro. Vol. 3 discusses various support and I/O chips. 895 pp. Vol. 2 [9L] $25.00, Vol. 3 [10Q] $20.00

Small Computer Systems Handbook

Sol Libes. The emphasis throughout this primer is on the important practical knowledge that the home computer user should have to be able to intelligently purchase, assemble, and interconnect components, and to program the microcomputer. Only a minimal knowledge of electronics is required to use this book. 196 pp. $8.45 [11D].

Accounts Payable and Accounts Receivable

Poole & Borchers. Includes program listings with remarks, descriptions, discussion of the principles behind each program, file layouts, and complete step-by-step instructions. Covers accounts payable and receivable in regard to invoice aging, general ledger, progress billing, partial invoice payments, and more. 375 pp. $20.00 [10V].

Microprocessors: From Chips to Systems

Rodnay Zaks. A complete and detailed introduction to microprocessors and microcomputer systems. Some of the topics presented are: a comparative evaluation of all major microprocessors, a journey inside a microprocessor chip, how to assemble a system, applications, interfacing (including the S-100 bus) and programming and system development. 416 pp. $9.95 [10S].

Programming in PASCAL

Peter Grogono. This book is an excellent introduction to one of the fastest growing programming languages today. The text is arranged as a tutorial containing both examples and exercises to increase reader proficiency in PASCAL. Contains sections on procedures, files, and dynamic data structures such as trees and linked lists. 359 pp. $11.50 [10A].

A Simplified Guide to Fortran Programming

Daniel McCracken. A thorough first text in Fortran. Covers all basic statements and quickly gets into case studies ranging from simple (printing columns) to challenging (craps game simulation). 278 pp. $13.50 [7F].

Some Common BASIC Programs

Poole & Borchers. This book combines a diversity of practical algorithms in one book: matrix multiplication, regression analysis, principal on a loan, integration by Simpson's rule roots of equations, chi-square test, and many more. All the programs are written in a restricted BASIC suitable for most microcomputer BASIC packages and have been tested and debugged by the authors. $12.50 [7M].

The Best of BYTE

This blockbuster of a book contains the majority of material from the first 12 issues of *Byte* magazine. The 146 pages devoted to hardware are crammed full of how-to articles on everything from TV displays to joysticks to cassette interfaces and computer kits. But hardware without software might as well be a boat anchor, so there are 125 pages of software and applications ranging from on-line debuggers to games to a complete small business accounting system. A section on theory examines the how and why behind the circuits and programs, and "opinion" looks at where this explosive new hobby is heading. $11.95. 386 pp. softbound from Creative Computing Press. [6F].

The Art of Computer Programming

Donald Knuth. The purpose of this series is to provide a unified, readable, and theoretically sound summary of the present knowledge concerning computer programming techniques, along with their historical development. For the sake of clarity, many carefully checked computer procedures are expressed both in formal and informal language. A classic series. Vol. 1: Fundamental Algorithms, 634 pp. $23.50 [7R]. Vol. 2: Seminumerical Algorithms, 624 pp. $23.50 [7S]. Vol 3: Sorting and Searching. 722 pp. $23.50 [7T].

BASIC With Business Applications

Richard W. Lott. This book focuses on the BASIC language and its application to specific business problems. Part One introduces the BASIC language and the concept of logical flowcharting. Part Two presents problems and possible solutions. Topics include: interest rate calculations, break-even analysis, loan rates, and depreciation. This book is a great aid to the beginner wanting to learn BASIC without having a technical or scientific background. 284 pp. $11.95 [10Z].

To order: send check or credit card number and expiration date (Visa, Master Card or American Express) plus $2.00 per order for shipping and handling to: S-100 MICROSYSTEMS, P.O. Box 789-M, Morristown, NJ 07960.

Running the 2 MHz S.D. Systems' ExpandoRAM at 4 MHz

by W. Howard Adams

The S.D. Systems' ExpandoRAM (the original 2 MHz version) is a quality product; the design examplifies that dynamic RAM can be of real benefit to a computer system. This design draws less current than any conventional static RAM design. The ExpandoRAM is 64K dense, eliminating the redundant, power-hungry TTL buffering and decoding circuitry necessary in multi-board memory systems. The board runs at 2 MHz with no wait states. It is clear that a great deal of forethought was put into the design since 8080, 8085 and Z80 processor timing differences have all been accounted for. S.D. Systems did a fine job in producing a super reliable 2 MHz card!

However, S.D. Systems could have gone further; the design has the potential to run at 4 MHz, with an added wait state. The purpose of this article is to save replacement costs of new memory by allowing 4 MHz operation through a simple modification.

The first and obvious requirement is to add a wait state to memory references. Refer to Figure 1 for a suggested implementation of a single wait state circuit. With very little time and effort it can be wire-wrapped in a kluge area, if a wait state circuit does not already exist in the system.

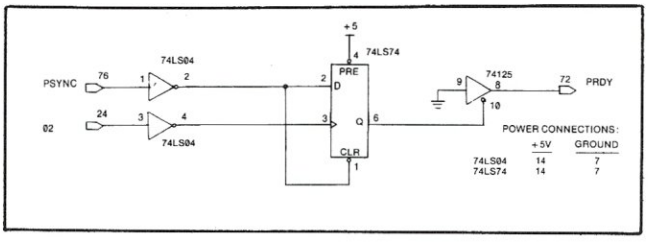


Figure 1
Adding One Wait State
(Every Memory Reference).

The second requirement is that the system's Z80 processor card must generate RFSH (refresh, active low) onto bus pin 66 which is used by the RAM board for refresh. Assure that on the ExpandoRAM card, E11 to E12 is jumpered and E10 to E12 is open. The ExpandoRAM's refreshing is now under the complete control of the Z80's timing.

W. Howard Adams, Eagles Computer Works, P.O. Box 22664, Denver, Co 80224

Before presenting the last requirement, some discussion of the board's only shortcoming is in order so that the modification will be better understood. Refer to the timing diagram in Figure 2 of this article, and to the schematic in the ExpandoRAM manual.

The problem is the generation of memory REFRESH CYCLEs which last much too long and run into the next M-cycle timing, completely obliterating it! As a fix was being sought, it was interesting to note how close to 4 MHz the board can work. While running a memory test there really were very few errors, and the addresses at which errors did occur were randomly located. This is a classic symptom of refresh interference!

The timing diagram of Figure 2 has one wait state in it, and is of an M1 cycle. Its inherent transparent REFRESH CYCLE occurs during the T4 state. Notice that the M1 and RFSH bus signals are for all intents and purposes the same signal, only the definitions of their active states are complemented. On the RAM board schematic observe that the RFSH signal eventually triggers one-shot U5-10, and a REFRESH CYCLE begins. U5-5 goes high, and U5-12 goes low throughout the REFRESH CYCLE, resulting in four occurances necessary to successfully refresh the RAMs.

First, U5-12 going low blocks the TIME-OF-CAS signal at U13-1 from being seen at U13-3 during a refresh. This permits the so-called RAS only REFRESH CYCLE, which is desirable.

Secondly, the active low at U5-12 also causes U8-8, 6, 11 and 3 to all go high enabling RAS drivers outputs at U3-8, 6, 11 and 3 to go low at TIME-OF-RAS. This produces the refresh RAS at all banks simultaneously, giving the RAM array its breath of fresh air.

Thirdly, U5-5 enables the refresh addresses of U11 and U15 to the RAM array by raising Select (pin 1) of the U21 and U16 multiplexers high.

Finally, after a time delay through R1 and C6 at U9-1, and onto U10-5, the active low leading edge at U10-6, defined as MEMORY CYCLE, triggers U5-1. Firing this one shot produces the TIME-OF-RAS signal, and a REFRESH CYCLE starts.

The timing diagram clearly differentiates between the M1 opcode fetch and the refresh portion. Note how long the REFRESH CYCLE waveform at U5-5 lasts past

New

S-100 A/D & TIMER

New

Tecmar's new A/D and Timer Board is designed to meet sophisticated data acquisition needs. The board can accommodate various A/D modules providing options such as 12, 14, 16 bit accuracy; 100 MHz throughput; variable ranges and gains. It contains a powerful timer circuit (AMD 9513) which can start A/D conversion and can also be used independently for time of day, event counting, frequency shift keying and many other applications.

TM-AD200 FEATURES

- Complies with IEEE S-100 specifications
- Transfers data in 8 or 16 bit words
- 30 KHz throughput standard
- 12 bit accuracy standard
- Jumper-selectable for 16 single-ended or 8 true differential channels
- External trigger of A/D
- Provision for synchronizing A/Ds
- Data overrun detection
- Data is latched providing pipelining for higher throughput
- Input ranges: ±10V, ±5V, 0 to +10V, 0 to +5V
- Output formats: Two's complement, binary, offset binary
- Auto channel incrementing
- I/O or memory mapped
- Utilizes vectored interrupt or status test of A/D
- Provision for expansion to 256 channels

TIMER FEATURES

- 5 independent 16 bit counters (cascadable)
- 15 lines available for external use
- Time of day
- Event counter
- Alarm comparators on 2 counters
- One shot or continuous frequency outputs
- Complex duty cycle and frequency shift keying outputs
- Programmable gating and count source selection
- Utilizes vectored interrupt

TM-AD200 OPTIONS

- Programmable gain up to 500
- 14 bit accuracy
- 16 bit accuracy
- Screw terminal and signal conditioning panel with optional thermocouple cold junction compensation
- 100 KHz throughput with 12 bit accuracy
- Low level, wide range (10mV to 10V FSR) permitting low level sensors such as thermocouples, pressure sensors and strain gauges to be directly connected to the module input

TECMAR INC.

TECMAR, INC.

(216) 382-7599

23414 Greenlawn • Cleveland, OH 44122

If your data acquisition needs are simple, the original Tecmar S-100 A/D Board will meet your needs.

$495

TM-AD100 FEATURES

- Complies with IEEE S-100 specifications
- 16 single-ended or 8 true differential channels
- 12 bit accuracy
- 25 KHz throughput
- I/O or memory mapped
- Input ranges: ±10V, ±5V, 0 to +10V, 0 to +5V
- Minimal software required.

For digital to analog conversion, Tecmar's D/A Board provides four independent 12 bit high speed D/A channels.

$395

TM-DA100 FEATURES

- Complies with IEEE S-100 specifications
- 4 independent digital to analog converters
- 12 bit accuracy
- 3 μsec settling time
- I/O or memory mapped
- Output ranges: ±2.5V, ±5V, ±10V, 0 to +5V, 0 to +10V

S-100 BOARDS

8086 CPU W/vectored interrupts	$450
RAM 8Kx16/16Kx8	$395
8086 PROM-I/O	$495
Serial and Parallel I/O	$350
Parallel I/O & Timer	$350

[1]Reg. Trademark of Tandy Corp.
[2]Reg. Trademark of Commodore

TRS-80[1] **PET[2]** **KIM[2]** **APPLE** *(available soon)*

A/D

- 12 Bit
- High Speed
- 8 Ch. Differential
- 16 Ch. Single-ended
- Each A/D Module $495

D/A

- 12 Bit
- High Speed
- 4 Channel

Each D/A Module $395

TRS-80 or PET expansion board, power supply, and enclosure $200.
Kim expansion board and power supply $150.

S-100 Real Time Video Digitizer

- Digitizes and Displays in 1/60 sec, flicker-free
- 16 Gray Levels
- Switch Selectable to display Black and White Graphics (8 pixels/byte)
- Maximum Resolution: 512 pixels/line x 240 lines
- Minimal software requirements

$850

Data Acquisition Systems and Video Microcomputer Systems Available

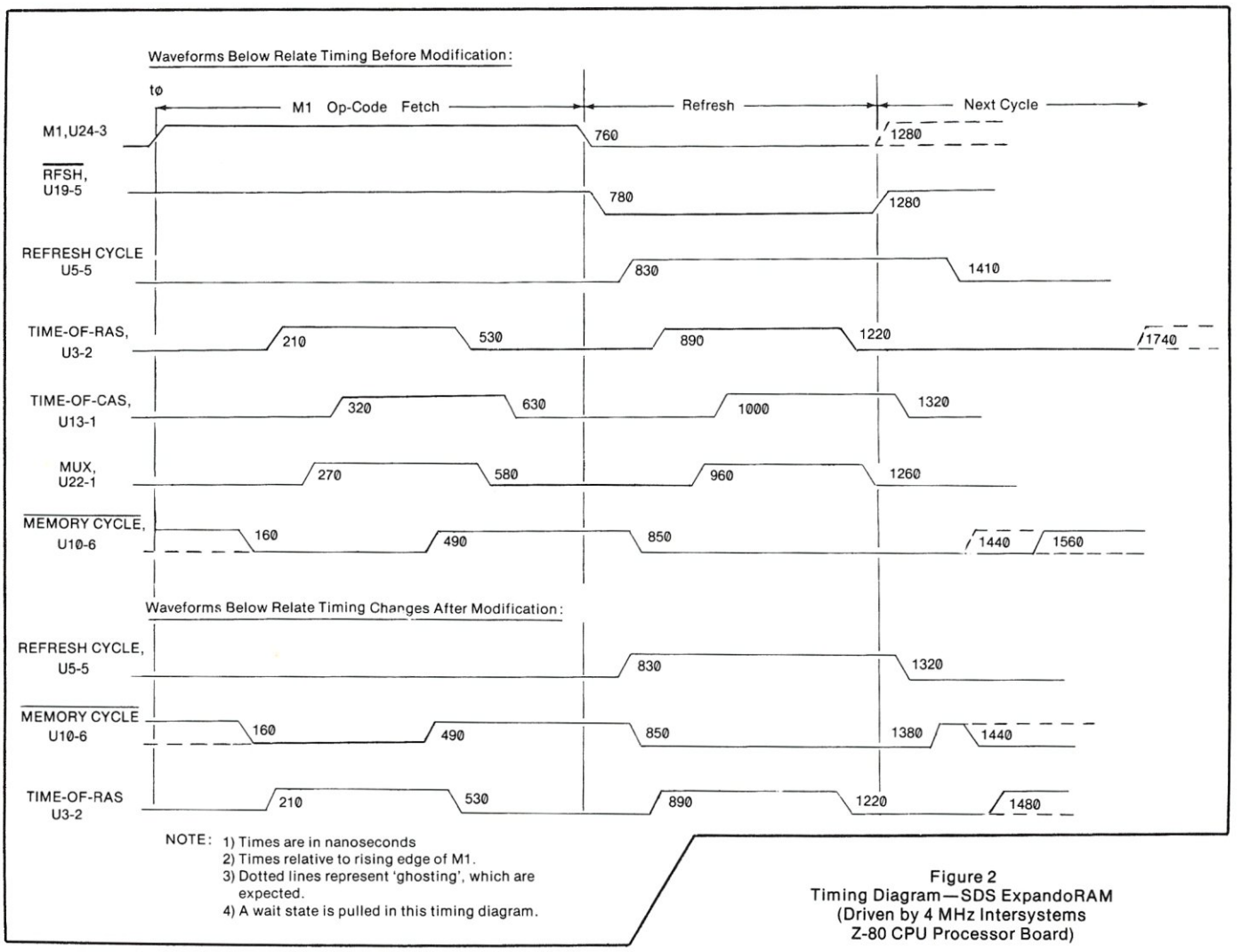


Figure 2
Timing Diagram—SDS ExpandoRAM
(Driven by 4 MHz Intersystems
Z-80 CPU Processor Board)

the alloted time for refresh! The culprit is: this signal lasts too long! As long as this signal is high, U10-6 is forced low, and no new cycle can begin and is therefore totally lost due to the synchronous nature of the S-100 Bus.

One quick fix could be shortening the REFRESH CYCLE timeout by reducing the value of R3 from 6.8K ohms. This is not a good method, since a one-shot does not provide consistent precise timing. There is an easier way which guarantees proper results without any component changing or trace cutting. Only one jumper is required. The proposed modification will also work should the board be returned to 2 MHz operation, so long as RFSH at pin 66 is provided.

The modification is simple. Pull U5 from its socket. Bend U5 pin 11 outward, and replace U5 back into its socket, assuring that all but pin 11 are in the socket. Run a jumper from U5-11 (which is hanging out) to U19-6. This jumper should be installed on the component side of the board, and dressed by running it under capacitors so it will not snag on neighboring pc boards during installation and removal.

A look at the timing diagram relates how the fix works. The new REFRESH CYCLE is cleared 100 nanoseconds sooner, and observe that the next M-cycle's trigger had been previously masked. The fix produces the necessary TIME-OF-RAS which had been lost. Refer to Figure 3 for the schematic changes necessary to correct the original.

The modification now keeps the REFRESH CYCLE one-shot cleared during non-refresh time. When it is time to do a refresh, the clear is removed, allowing the one-shot to fire. At the end of the REFRESH CYCLE, as defined totally by the bus signal RFSH, the

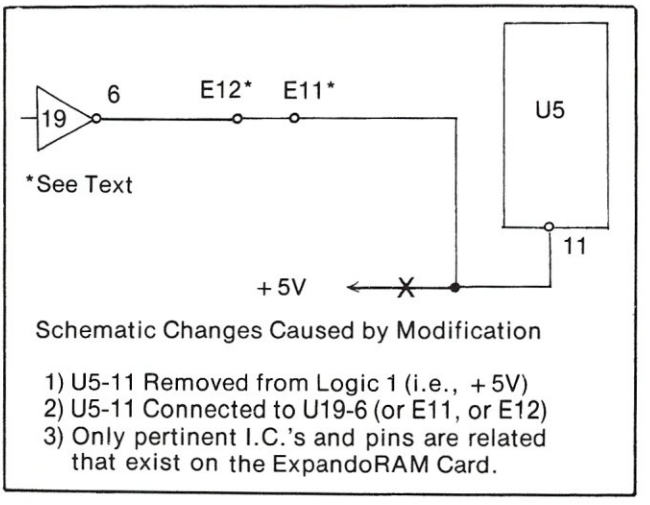


Figure 3
Modification
Schematic—SDS ExpandoRAM

modification clears the one shot efficiently and asynchronously.

The 2 MHz ExpandoRAM should now function at 4 MHz with its required wait state without sacrificing system throughput too drastically. (The particular board used to find this fix is populated with 250 nanosecond RAMs, and the only wait state needed is during the shorter accessing of the M1 cycle. So this system is hardly being held back at all!)

So go ahead and spend that memory replacement money on the new peripheral you've wanted for so long. There's still plenty of life in the 2 MHz ExpandoRAM with this new 4 MHz system! □

Pascal Speed Comparisons

by Fred Greeb

How fast is Pascal? Current advertisements for UCSD Pascal state "A UCSD Pascal program executes up to ten times faster than a comparable BASIC program". Since I recently acquired UCSD Pascal this statement led me to conduct a speed comparison between Pascal and the other programming languages available to me.

The four languages used for the comparison were Microsoft Extended Basic, version 4.51; CBasic-2, compiler version 2.01, interpreter version 2.03; Microsoft Fortran, version 3.2; and UCSD Pascal, version II.0. With the exception of Fortran, these languages are all interpreters, operating on an intermediate code which is either generated directly as the program is entered (for Microsoft Basic), or generated by a separate compilation process. Fortran operates as a true compiler generating native machine code.

Four programs were used for the test. These programs are presented in Table 1. The first program is a simple do-nothing loop; the second includes addition, multiplication, and division with in the loop; and the third adds a subroutine call within the loop. The fourth program adds the process of raising a number to a power. Since my UCSD Pascal manual states that there is no explicit provision for this process, it is handled as a programmed function for all four languages.

The results of the test are shown in Table 2. All programs were run on my Altair 8800 system, which uses an 8080 microprocessor with a 2 Mhz clock. The execution times were measured between the printout of the "start" and "done" messages within the programs, using the seconds display of a digital watch. Each program was run three times, with the average time presented in the table, to remove potential timing errors.

	Execution time in seconds			
Language	Program A	Program B	Program C	Program D
Microsoft Basic	7	13	14	63
CBasic-2	28	51	52	653
Microsoft Fortran	1/2	7	8	53
UCSD Pascal	6	7	8	103

Table 2. Timing Comparison Results

I am not going to attempt to interpret the results of this test, but the following observations do seem appropiate. First, I have only had access to Pascal for a few weeks and am not yet very familar with its operation. Therefore, I chose simple programs (and program structures) which could easily be programmed similarly in the four languages. Second, these tests primarily compare the floating point arithmetic capabilities of the four languages, rather than any other of the capabilities or features they may possess. Since this is a common feature of many of the applications I am involved in, it is an area of primary interest to me. The poor showing of CBasic in this application can be partially attributed to the fact that it carries 14 digit precision for all real variable operations. To achieve this same precision with the other languages would require double precision variables, which would undoubtably slow them down.

A more thorough timing comparison of Pascal with other languages would be of interest to me. However, this project will have to wait either until I become more familar with Pascal (and have time to do the comparisons), or until someone else undertakes such a project. □

Fred Greeb, 1915 S. Cape Way, Denver, CO 80227

Program A

Basic

```
100 DEFINT K
200 PRINT"START"
300 FOR K=1 TO 10000
400 NEXT
500 PRINT"DONE"
600 STOP
700 END
```

CBasic

```
PRINT"START"
FOR K% = 1 TO 10000
NEXT
PRINT"DONE"
STOP
END
```

Fortran

```
      WRITE(1,1000)
      DO 100 K = 1,10000
100   CONTINUE
      WRITE(1,1010)
      STOP
1000  FORMAT(1X,'START')
1010  FORMAT(1X,'DONE')
      END
```

Pascal

```
PROGRAM BENCHMARK;
VAR K:INTEGER;
BEGIN
  WRITELN('START');
  FOR K := 1 TO 10000 DO
    BEGIN
    END;
  WRITELN('DONE');
END.
```

Program B

```
100 DEFINT K
200 X=33.24
300 PRINT"START"
400 FOR K=1 TO 1000
500 A=X/7.5 + X*6.2
600 NEXT
700 PRINT"DONE"
800 STOP
900 END
```

```
X = 33.24
PRINT"START"
FOR K% = 1 TO 1000
  A = X/7.5 + X*6.2
NEXT
PRINT"DONE"
STOP
END
```

```
      X = 33.24
      WRITE(1,1000)
      DO 100 K = 1,1000
        A = X/7.5 + X*6.2
100   CONTINUE
      WRITE(1,1010)
      STOP
1000  FORMAT(1X,'START')
1010  FORMAT(1X,'DONE')
      END
```

```
PROGRAM BENCHMARK;
VAR K:INTEGER;
    X,A:REAL;
BEGIN
  X := 33.24;
  WRITELN('START');
  FOR K := 1 TO 1000 DO
    BEGIN
    A := X/7.5 + X*6.2;
    END;
  WRITELN('DONE');
END.
```

Program C

```
100 DEFINT K
200 X=33.24
300 PRINT"START"
400 FOR K=1 TO 1000
500 A=X/7.5 + X*6.2
550 GOSUB 850
600 NEXT
700 PRINT"DONE"
800 STOP
850 RETURN
900 END
```

```
      X = 33.24
      PRINT"START"
      FOR K% = 1 TO 1000
        A = X/7.5 + X*6.2
        GOSUB 100
      NEXT
      PRINT"DONE"
      STOP
100   RETURN
      END
```

```
      X = 33.24
      WRITE(1,1000)
      DO 100 K = 1,1000
        A = X/7.5 + X*6.2
        CALL DUMMY
100   CONTINUE
      WRITE(1,1010)
      STOP
1000  FORMAT(1X,'START')
1010  FORMAT(1X,'DONE')
      END
      SUBROUTINE DUMMY
      RETURN
      END
```

```
PROGRAM BENCHMARK;
VAR K:INTEGER;
    A,X:REAL;
PROCEDURE DUMMY;
  BEGIN
  END;
BEGIN
X := 33.24;
WRITELN('START');
FOR K:= 1 TO 1000 DO
  BEGIN
    A := X/7.5 + X*6.2;
    DUMMY;
  END;
WRITELN('DONE');
END.
```

Program D

```
100 DEF FNP(E,X) = EXP(X*LOG(E))
200 DEFINT K
300 X = 1
400 Y = 1.57
500 PRINT"START"
600 FOR K=1 TO 1000
700 Z=FNP(X,Y)
800 X=X+1
900 NEXT
1000 PRINT"DONE"
1100 STOP
1200 END
```

```
DEF FN.POWER(E,X) = EXP(X*LOG(E))
X = 1.0
Y = 1.57
PRINT"START"
FOR K% = 1 TO 1000
  Z = FN.POWER(X,Y)
  X = X + 1.0
NEXT
PRINT"DONE"
STOP
END
```

```
      X = 1.0
      Y = 1.57
      WRITE(1,1000)
      DO 100 K = 1,1000
        Z = POWER(X,Y)
        X = X + 1.0
100   CONTINUE
      WRITE(1,1010)
      STOP
1000  FORMAT(1X,'START')
1010  FORMAT(1X,'DONE')
      END

      FUNCTION POWER(E,X)
        POWER = EXP(X*ALOG(E))
      RETURN
      END
```

```
PROGRAM BENCHMARK;
VAR X,Y,Z:REAL;
    I:INTEGER;
FUNCTION POWER(E,X:REAL):REAL;
BEGIN
  POWER := EXP(X*LN(E));
END (* POWER *);
BEGIN
  X := 1.00;
  Y := 1.57;
  WRITELN('START');
  FOR I := 1 TO 1000 DO
    BEGIN
    Z := POWER(X,Y);
    X := X + 1.0;
    END;
  WRITELN('DONE');
END.
```

Table 1. Programs Used for Comparison

The ADS Noisemaker Board

by Steve Levine

PART-1 Building and testing this dual programmable sound generator kit.

The Ackerman Digital Systems NOISMAKER is an S-100 standard board incorporating 2 General Instruments AY3-8910 integrated circuits. The 8910 is a large scale integrated circuit which can produce a wide variety of complex sounds under software control. Once the internal registers have been programmed, sound will continue to be produced, allowing the processor's time to be used for other tasks.

KIT DESCRIPTION

Construction of this kit was quite simple. It should pose no problem for the novice or experienced hobbyist. Adequate documentation was included with the board consisting of a brief overview of the 8910 chip, clear assembly instructions and a good parts list. Using only the schematic diagram and parts list supplied, I had it going in a little over an hour. The circuit board is good G-10 solder-masked plated through construction.

Included on the board is a 1 watt audio amp, which can be connected directly to a speaker for output. The edge fingers for both the bus and interface connectors are gold plated. A very nice feature of the board is the plated through 'kluge' area provided on the right hand side. I found this particularly handy for adding a second audio amplifier chip or anything my imagination could muster.

Provisions are made for the 2 Mhz system clock to be used as the time base for the synthesizer. There is also a custom clock prototype area with enough room for the 3.57945 mhz crystal clock circuit that G.I. calls for in the data sheet. (This frequency facilitates using an equally tempered scale [ETS] set of numbers for the tone generators.) This area is supposed to be designated for adding a wait state generator, but will serve in the above capacity just fine.

Note: The manufacturers of the board advise using wait states with a 4 Mhz system (this is an inherent limitation of the AY3-8910 IC).

Steve Levine, 907 East Butler Ave., Ambler, PA 19022

Another feature of the PSG is the on board i/o ports. ADS brings three of the four 8 bit ports out to the edge connector on top of the card. The fourth one is routed to the kluge area. In computer music experimentation you will undoubtedly find a use for these lines, say for scanning a keyboard to facilitate an input device.

OPERATION

The board is addressed in the I/O map of a standard S-100 system. It occupies 4 concurrent I/O addresses. Using the dip switch -- select the starting block of four I/O addresses you want the board to appear in. I chose 0 Hex for simplicity.

Address bit	0	1	Looking at:
< ALADRA >	0	0	Latch address for psg 'a'
< ADATA >	0	1	Register data for " 'a'
< BLADRA >	1	0	Latch address for " 'b'
< BDATA >	1	1	Register data for " 'b'

By writing the desired register number to the ALADRA or BLADRA I/O port on the noismaker board, and then writing the value to the ADATA OR BDATA I/O port, simple tests can be performed on all the different registers.

The ADS NOISEMAKER takes very good advantage of the AY3-8910 for the S-100 bus. With an assembly language driver and a good high level language such as PASCAL or FORTH, one can easily implement a tiny music composition and performance system. The 8910 is not a complicated device to use, but a good understanding of the PROGRAMMABLE SOUND GENERATOR will of course be necessary. Refer to architecture chart and diagram for the following discussion:

The AY3-8910 Programmable Sound Generator is a 3 channel digitally programmable synthesizer in one IC. There are 16 internal registers which are individually addressed , into which tone, envelope and control information are loaded prior to producing a sound. Included on chip

are register decoding and data bus buffering circuitry. Upon a RESET (negative) all registers are reset to zero.

Each chip contains the following :

* 1 I/O enable, mixer control register
* 3 - 12 bit tone generators
* 3 - 4 bit amplitude control registers
* 1 - 5 bit noise generator
* 1 - 16 bit envelope generator
* 1 - 4 bit envelope control register
* 2 - 8 bit parallel I/O ports

Individual registers can be set to specific values for producing sound. Refer to architecture chart. There are 3 individual 12-bit tone generators. When a 12-bit number is loaded into a set of tone period registers, the value is then latched. The clock input is divided by 16 and again divided down by a decimal value equal to the tone period registers. When the counter reaches zero, the register containing the original number is reloaded into the counter and the process is repeated. The result is a continuous square wave tone , whose period is a function given by the following equation :

TONE FREQUENCY ft= clock/(16TP)

where:
clock = frequency of clock to AY3-8910
TP = 256* coarse tune + fine tune
coarse tune = Coarse tune register value
fine tune = Fine tune register value

(Numbers are assumed decimal)

Register 7 is used to control the selection (mixer) of either tone or noise or both, for a particular channel. Registers 8, 9 and 10 are amplitude control registers. By setting the mode bit low, the processor can directly control amplitude of a channel with the remaining 4 least significant bits of these registers. With the mode bit high, envelope control is passed to the envelope generator section.

To produce a simple sound you would first set the mixer to the desired configuration, then using the amplitude register correspnding to that particular channel, set the amplitude level from 0 to 15 levels of loudness. When noise is selected on any or all of the channels the above formula can be similarly applied to calculate the 5 bit noise period as follows:

NOISE FREQUENCY fN = clock/ 16 NP

where:
NP = 5 bit noise period value

For an envelope period calculation:

ENVELOPE FREQUENCY fE = clock/ 256 EP

where:
EP=256 Coarse envelope period+Fine envelope period
clock=frequency of clock to AY3-8910

Using the above formulas, and a 2 Mhz clock, we can obtain a usable range of tone frequencies from 30.5 hz. to 125 Khz. With the noise generator period value we obtain a noise range of 4 Khz. to 125 Khz. The envelope generator has the more sophisticated control features of the PSG. Envelope period can be calculated to 16 bits resolution. A usable range for the envelope period would be 0.12 hz to 7812.5 hz. This period will control the frequency of the envelope generator. Envelope shape/cycle (4 bits) provides a function control for the envelope generator. Referring to the envelope shape diagram, HOLD, ALTERNATE, ATTACK, and CONTINUOUS bits control envelope generation in conjunction with the envelope period registers. This particular function could be discussed in endless detail, but the manufacturers of the 8910 have greatly simplified this with the envelope shape diagram. Experimentation with this module is probably the best way to fully understand it's capabilities.

In a future article, I will discuss software and hardware for a complete music system, which is written in FORTH. A musical keyboard with polyphonic and velocity sensitive capabilities will be used as the musical input device.

I consider the ADS NOISEMAKER is an ideal and inexpensive way for an S-100 system owner to experiment with computer music.

I welcome correspondance related to this article and computer music generally. □

Evaluating the New 8088 Microprocessor

This article compares the new Intel 8088 8-bit microprocessor to the Motorola 6809 and several other processors. This report was furnished by a source within Intel who requested that his name not be disclosed. He did however give permission to reprint the report. Some editing was done on the report to improve readability.

Editor

8088 ASSEMBLY LANGUAGE BENCHMARKS

Following is a summary of 8088 implementations of the four benchmarks analyzed in the article entitled 'Assembly Language Benchmarks' written by A. Flippin, which appeared in the March 1980 issue of KILOBAUD magazine. The reader should refer to the actual article for a detailed explanation of each benchmark and the criteria used for evaluation.

The 8088 is compared against the following machines (listed in order of overall performance ranking).

370-145
LSI-11
9900
Z80
6502
6800
8080
6100
1802
SC/MP

The new ranking now is:

8088
370-145
LSI-11
9900
Z80
6502
6800
8080
6100
1802
SC/MP

All timings are calculated for the standard speed (5mhz) 8088. The **8mhz** version, expected about the middle of '81, will provide even better performance!

Intel claims therefore that "The 8088 is clearly the highest performance 8 bit microprocessor in the world!"

Benchmark - Table look up
Number of instructions- 8 rank - #1 (tie 370,LSI-11,9900)
Memory utilized (bytes)- 20 rank - #2
Execution speed (uS) - 58.8 rank - #1

Benchmark - Block move
Number of instructions- 11 rank - #1 (tie 370,LSI-11,9900)
Memory utilized (bytes)- 23 rank - #2 (tie 8080)
Execution speed (uS) -1075.6 rank - #2

Benchmark - Jump Table
Number of instructions- 3 rank - #1 (tie LSI-11)
Memory utilized (bytes)- 10 rank - #1 (tie LSI-11)
Execution speed (uS) - 10.2 rank - #2

Benchmark - 16 bit multiply
Number of instructions- 2 rank - #1 (tie 370,LSI-11,9900)
Memory utilized (bytes)- 8 rank - #1 (tie 370,LSI-11,9900)
Execution speed (uS) - 16 rank - #3

Figure 1

Averages by Benchmark

Table Lookup		Block Move		Jump Table		Multiply	
rank	index	rank	index	rank	index	rank	index
#1	.66	#1	.71	#1	.56	#2	.2

Figure 2

Averages by Category

Execution Time		Ease of Programming		Memory Utilization	
rank	index	rank	index	rank	index
#1	.47	#1	.490	#1	.64

Overall	
rank	index
#1	.53

Figure 3

8088 Versus 6809

In an effort to determine which microprocessor actually delivers the most performance, the Intel 8088 was compared to the Motorola MC6809. Nine programs

were written, and the two processors were compared on the basis of execution speed, ease of programming (number of lines of code), and memory efficiency. The results in the table below show that the 8088 significantly outperformed the MC6809 in all three catagories.

PROGRAM	SPEED 6809/8088	LINES 6809/8088	BYTES 6809/8088
Computer Graphics	23.44	5.80	4.50
16-Bit Multiply	2.01	7.00	4.00
Vector Add	1.10	1.00	1.17
Reentrant Call	.87	.88	1.02
Computed Jump	.64	1.17	.93
Block Move	2.06	2.00	1.73
Block Translate	1.78	1.30	1.54
Character Search	2.12	1.13	1.06
Word Shift	3.09	2.25	1.80
Normalized Average	4.12	2.50	1.97
Adjusted Average	1.86	2.09	1.76

Figure 4

The normalized averages show that the 8088 was more than four times faster than the MC6809, required 2.5 times less lines of code, and was nearly 2 times as efficient in memory usage. The normalized averages were greatly influenced by the Computer Graphics benchmark which requires many 16-bit multiplies and divides. For this reason an adjusted average was computed ignoring the highest and lowest value in each catagory.

A brief description of the benchmark programs follows:

COMPUTER GRAPHICS...Scales the X and Y values of an array for a graphics display.

16-BIT MULTIPLY.....Multiplies two unsigned 16-bit numbers to produce a 32-bit product.

VECTOR ADD .....Adds the elements of two 20 element vectors and stores the resultant vector in memory

REENTRANT CALL .....Calls a subroutine, saves all registers, adds paramaters passed to the subroutine and returns after restoring all registers.

COMPUTED JUMP .....Uses a control byte and a vector table to compute a jump address.

BLOCK MOVE.....Moves a block of data in memory to another location in memory.

BLOCK TRANSLATE ...Translates a block of EBCDIC characters to ASCII.

CHARACTER SEARCH....Searches a block of memory for a character.

WORD SHIFT.....Shifts a 16-bit word 5 places to the right.

These benchmark programs demonstrate the ability of the processors to perform many of the tasks typically required of a high performance microprocessor. The 8088 performs so well here due to its powerful instruction set which includes 8- and 16-bit signed and unsigned multiplies and divides, and string processing instructions with optional repeat prefixes. The register set of the 8088 is also extremely versatile with four 16-bit accumulators which can be used as eight 8-bit registers, four pointer registers, and four segment registers. The MC6809 has only an unsigned 8-bit multiply and no divide. It has a 16 bit register which can be used as two 8-bit accumulators or one 16-bit accumulator. The MC6809 also has four pointer registers.

The Z80 was also looked at, and was out performed by more than three to one by the 8088 in the adjusted average for execution speed.

These results clearly demonstrate that the Intel 8088 outperforms the Motorola MC6809 in the catagories of speed, ease of programming, and memory efficiency. The 8088 also provides a clear upgrade path to 16-bits with complete software compatability with Intel's 16-bit 8086.

Attention S-100 Board Suppliers

We are currently attempting to compile a listing of all S-100 board and mainframe suppliers and their products. If you are a manufacturer of S-100 products please send us a complete set of specification sheets on your products. We hope to publish this listing in the NOV/DEC issue of S-100 MICROSYSTEMS.

Relocatable Code

by Richard H. Mossip

WHAT IS IT?... Relocatable code is a program which can be run correctly when located at any memory address. In some computers it can be written directly, as all memory accesses are processed through an indexing scheme where a pre-defined constant is added to each memory address to obtain the address used in real memory. This is a hardware approach used on many large machines where the actual address of any program will be different each time it is run, depending on what is in the machine with it in a multi-programming setup. All addresses in a program are logical, and are mapped by hardware and the operating system into physical addresses used at the time. It is much more difficult in micros, particularly the 8080 & 8085, as they lack an indexed addressing mode (which the Z-80 has but is rarely used).

HOW DO YOU DO IT?... In the 8080, and other simple processors which have only direct addressing (at a specified 16 bit address) or register indirect addressing (at the content of a specified register-e.g. LDAX B), when a program requires to run at a different location all 2-byte addresses located within the code module (particular program segment to be relocated) have to be changed to reflect the new location, so that all jump and call statements have the correct destination, and all addresses for data are correct. There are two approaches generally used for this. The first is usually used for "Page Boundary relocation", where a program can be relocated in 256 byte increments, and only the upper byte of addresses is changed. Conceptually the same approach could be used for full relocation, but this is rarely done as it is more complex. This can be performed with a standard assembler by making two assemblies; the first ORGed at 0000 and the second at 100 Hex. These two are then compared, and a list is made of all locations where a difference exists. These are the locations which have to be patched with the correct page location where the program will reside. The page boundary at which the program starts is added to the contents of each location in the list, and the program will now run without any special code.

Richard H. Mossip, 13 W Buena Vista Way, Bloomingdale, NJ 07403

The second method uses a special assembler called a relocating assembler. Examples of this are the TDL relocating Macro assembler, and the Microsoft ASM80. In these the object code file produced is different from that produced by a conventional assembler in that all addresses generated by a label within the program are flagged for relocation, while all addresses given as constants are assumed to be fixed. The program is loaded into working memory by a special "linking loader" which adds the appropriate constant to the flagged addresses (which reference the beginning of the program as their base) and produces properly running code which can start at any address in memory.

WHY BOTHER?... Two reasons come to mind which make this useful in a microprocessor situation. The first occurs in CP/M, where the operating system resides at the top of memory in use, and is different for different system sizes.

System sizes usually only differ in 1K or more increments, so page boudary relocation is adequate, and Digital Research only wishes to distribute object code, thus avoiding the simple approach of re-assembling with a new origin. There is a program called MOVCPM distributed with CP/M which contains a table of all locations requiring change, and accepts console input of the new system size. This builds in memory a copy of the code with the new addresses. It is then written out to disk as the new system, after having the user-provided parts of the program suitably re-assembled and grafted onto it with the debugger. This is a process which requires a certain amount of operator attention and knowlege, but works well in the application where it seldom needs to be done.

Where a program must be run in minimum memory, as in special purpose industrial applications, it is very convenient to write the program in modules which can be tested and debugged separately. Once a module is debugged, one does not want to compile or assemble it again, to avoid introducing errors in patching several short segments together to make a long program. Relocatable code will do this for you. It also enables you to write code in Basic or Fortran, debug it and then re-code it a module at a time in assembly

language, testing as you go-just linking them together as they are written. If the length of one module is changed, the required changes to the others are made entirely automatically at load time. This is the approach used by the Microsoft ASM80, which produces code directly compatible for linking with Basic Fortran or Cobol programs. The format is totally different from that used by TDL (a modification of the Intel checksum HEX loader) as the code is only stored on disk as a bitstream (not located on BYTE boundaries) to conserve disk space. This is at least a 2:1 compression, and probably better. Details are given in the manual (barely) and are of little interest to others.

GILDING THE LILY..... This was clearly too simple, so the designers rushed out to make it more powerful (e.g. complicated). The concepts of INTERNAL, EXTERNAL, and ENTRY labels are needed, if programs are to be linked together. An address flagged to the assembler as EXTERNAL means that it is defined in another module (and must NOT be redefined in this one). An INTERNAL address label is flagged as available for access by other external programs. An ENTRY label is an internal label at which the program can be entered, rather than referenced. This is used when modules are placed in a library, and is a name for which the library can be searched in the TDL system. All other labels are for internal consumption only and can be duplicated in other program segments without fear of confusion.

More details will be given in a later article as I become more familiar with using the system. □

NEW PRODUCTS

EXPANDORAM MOD FOR 4MHz OPERATION

J.E.S. Graphics, Box 2752, Tulsa OK, 74101 (tel: 918-742-7104) is providing a modification kit (4 parts) which when installed in an ExpandoRam Memory Board enables it to operate at 4MHz, with no wait states. Cost is $10.00

DESKTOP PASCAL GRAPHICS COMPUTER SYSTEM

The demand for lower-cost graphics capabilities has resulted in a new product from Integrated Research and Information Systems (IRISystems) Corporation, the ENSEMBLE I20GX. The new product is a self-contained computer system, packaged in a configuration that might easily be mistaken for a terminal. Standard versions are based upon the Western Digital Pascal MicroEngine, high resolution graphics, a 15" monitor, 12-slot S-100 standard motherboard, detachable keyboard, double density, double-sided dual floppy disk subsystem, Z80 alternate on-board MPU, memory parity, 128KB dynamic RAM, UCSD Pascal, CP/M, constant voltage transformer, and printer port for graphics hard copy output. The ENSEMBLE I20GX can present a 768 X 480 pixel format in 8 X 16 pixel characters; some of these can be user-defined. Single quantity price is $9796.

The ENSEMBLE I20GX in its standard configuration runs under UCSD Pascal and CP/M. Alternate configurations run only under CP/M, do not use the MicroEngine, but offer a Pascal compiler which generates floating point instructions for the AMD9511 chip. A low-cost version of the ENSEMBLE I20GX uses the Z80 MPU, a 9" black and white monitor, and one 5.25" drives. It costs $3464 in single quantity.

For more information write: Integrated Research & Information Systems, 10150 Sorrento Valley road, suite 320, San Diego, CA 92121, (714) 457-3730.

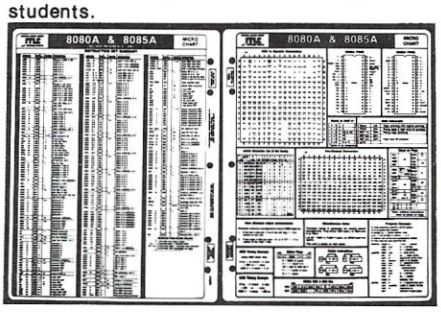
S-100 8088 CPU BOARD ANNOUNCED

ACOM Electronics has released its model P188 8088-based S-100 CPU board. The P-188 will run in either of three modes: 1) As a stand-alone processor on S-100 bus, 2) As a slave processor with other processor cards, and 3) with one or more additional processors in a shared processor environment. The P188 emulates all necessary S100 Bus signals. Numerous jumpers allow configurating the card to run different operating modes, as well as static and dynamic memory.

The 8088 processor is an 8086 with an 8 bit data bus. It has 16 bit internal architecture, addresses 1 million bytes of memory, and 8-bit/16-bit signed and unsigned arithmetic in binary or decimal, including multiply and divide. The introductory price is $385, Tested & Assembled. For more information contact: Sam R. Hawes, jr, ACOM Electronics, 4151 Middlefield Rd, Polo Alto, CA 94303; Tel: (415) 494-7499.

8080/8085 PLASTIC ACTION SHEET

Micro Chart is a multiple color plastic sheet packed with instant access technical information. #100A covers the 8080 / 8085 microprocessors and is the first in a series. Made from credit card type plastic, it measures 8½" by 11" and is puched for optional notebook use. Primarily software oriented, it also includes critical hardware data. It is organized to save time for professionals, hobbyist, and students.



Micro Chart is available at many dealers for only $2.95. Dealer inquiries invited. for mail orders to Micro Logic add $1.00 for P&H. shipment is 5 days ARO. Micro Logic Corp. Dept. CSL POB 174, Hackensack, NJ 07602, (201) 342-6518

TRS-80 CASSETTE INTERFACE FOR S-100 SYSTEMS

TRS-80* programs on cassette can be loaded into an S-100 system using a hardware interface and software driver from J.E.S. Graphics, Box 2752, Tulsa OK, 74101 (tel: 918-742-7104). The hardware is assembled. Documentation is provided which tells the user how to relocate loaded programs to the correct address, find the entry point and link the program to keyboard input and video output routines. Included is an example of how to interface SARGON-II* and TRS-80* Level-II Basic to an S-100 system. Price is $30.00. *Sargon-II and TRS-80 are registered trademarks of Hayden Book Company and Landy Corp., resoectively.

VERSATILE CPU AND I/O CARD WITH DISK CONTROLLER

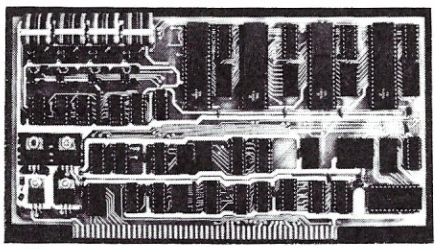
The CP/IO-I, from Arkon Electionics, provides on a single S-100 card, all of the CPU and I/O facilities required to construct a disk-based microcomputer system. This 8080A based card features fully vectored interrupts, 5 programmable interval timers, 24 parallel I/O lines, RS-232 serial terminal port at 110 baud to 76 kilobaud, and an RS 232 pseudo-serial port suitable for printer interface. On-board EPROM (2708 or 2716), with power on vectoring, allows user memory to reside from 0000H to DFFFH.

Disk I/O supporting IBM 3740 soft sectored format is provided for up to four 8" or 5¼" drives and auxilliary software driven cassette. All resident I/O devices may be accessed either in I/O space or as memory locations allowing for optimal program I/O access. A CP/M bootstrap and BIOS EPROM (complete with source listing) is included with the standard system and custom configurations are available on special order. Despite its high functional density the CP/IO-1 typically requires only 8V at .8A, +15V at .15A and -15A and -15V at .05A.

Priced at $499.00, the CP/IO-1 is available from: ARKON ELECTRONIC 409 Queen Street West Toronto, Ontario.

4-CHANNEL SERIAL I/O FOR S-100 SYSTEMS

Theta Laboratories, a maker of small computer systems for process control, automatic testing, and other special applications, has announced the first of a family of specialty I/O and peripheral control boards.



The S10-4 is well-suited for multiple-terminal systems, of any other application requiring several serial connections, RS-232 or current loop. Four UARTs make the channels independent for Baud rates of 75 to 9600. An on-board crystal oscillator makes Baud settings independent of the CPU clock. The price is $249.50.

For more information contact: Theta Laboratories, Inc. 10911 Dennis Rd. #405 Dallas, Texas 75229 phone 214-241-1090.

MULTI-BANK MEMORY CARD

The Digiac MAPS-1001 memory card is compliant with the IEEE-100 bus standards for extended memory bank addressing. It has the following features:

— Two independent memory segments. Typical configuration—2 segment by 32K bytes per segment, each origined from OH to 7FFFH.

— Single memory segemtn 16K to 64K selectable.

— 375ns memory access and 500ns memory cycle time.

— On-board memory bank address decoder-Dip switch selectable.

— Utilizes 4116 memory technology.

— Power dissapation less than 6 watts.

For more information contact: Digiac Corp., 175 Engineers Road, Smithtown N.Y., 11787; tel:(516) 273-8600.

SINGLE BOARD COMPUTER (SBC)

The Digiac CT-804 is a complete Z80A computer system, on a S-100 card, offering the features:

— Three (3) independent bi-directional EIA serial channels with selectable data size, parity, and baud rate options. Each channel is supported by a 8251 USART device.

— Two (2) independent TTL 8 bit parallel output channels with one and one half (1½) TTL 8 bit input channels.

— Power on/reset to on-board Digiac system monitor.

— Powerful 3K byte on-board-Digiac system monitor.

— 1K byte RAM.

— Memory management logic conforming to IEEE standards-up to 1 megabyte of memory.

— Fully operational vectored interrupt logic.

— I/O byte configuration-Dip switch or Digiac monitor selectable.

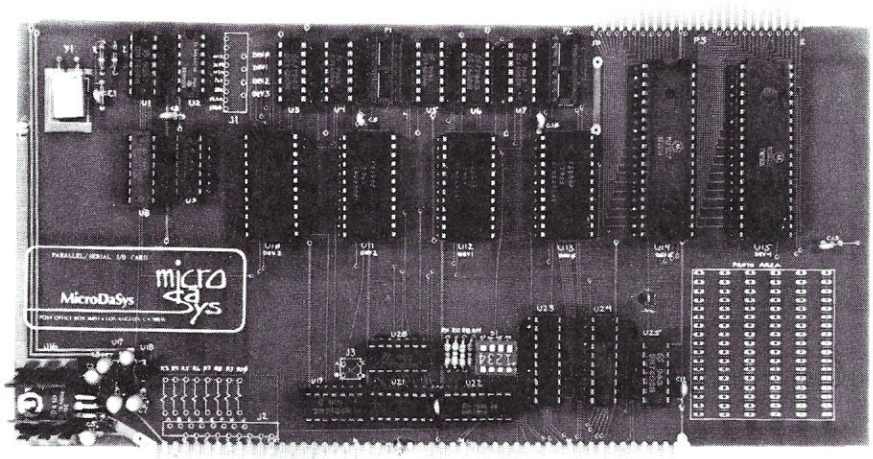
— 2 MHz or optional 4 MHz operation.

For more information contact: Digiac Corp., 175 Engineers Road, smithtown, N.Y., 11787: tel: (516) 273-8600.

THE CHANNEL PARALLEL/SERIAL I/O CARD AVAILABE

MicroDaSys has introduced the 4P4S combines four parallel bi-directional data ports (32 I/O bits) with full handshaking and interrupt control (another 8 I/O bits) for S-100 systems. In addition there are four serial RS-232 input and output ports. These serial ports also operate under full handshaking and interrupt control. One portion of the board is pre-drilled and plated through for use as a proto-typing area for custom applications.

Price of the 4P4S is $199 in kit form and $299 completely assembled and tested and is available from MicroDaSys, P.O. Box 36215, Los Angeles, CA 90036, phone (213) 731-0876.



MP/M SUPPORT CARD

The Digiac MAPS-1000 MP/M* Universal support Module is an S-100 card designed to meet the demands required by Digital Research's MP/M multi-tasking operating system. All input/output, interrupt generation for task switching, and disk bootstrapping are resident functions on the MAPS-1000.

The MAPS-1000 is totally compatible with the IEEE S-100 bus standards. It provides the business professional or systems designer with a simple and effective way to rapidly install and exploit the total benefits of MP/M.

The MAPS-1000 has the following:

— Four (4) independent EIA RS-232C serial data channels w/jumper selectable data bits, start/stop bits, parity select, and baud rate up to 19.2KBPS.

— Two (2) TTL parallel input/output channels, each with strobe-acknowledge capability, and latched or transparent output.

— On-board phantom bootstrap prom memory w/jumper selectable enable/disable.

— High accuracy crystal controlled interrupt generation capability. Required by MP/M for task switching and time of day/day of week calculations. Utilizes a proprietary interrupt count correcting circuit to insure accurate time calulation for scheduler and TOD functions. Interrupt rate selectable for either 30 or 60 Hz.

— Software selectable restart vector mode-supports all 8080 and Z80 interrupt operations.

Complementing the MAPS-1000 are Digiac developed MP/M xios packages that support several single and double density S100 disk controllers. The Digiac developed xios packages have been preconfigured to operate exclusively with the MAPS-1000. For more information contact: Digiac Corp., 175 Engineers Road, Smithtown n.y., 11787; tel:(516) 273-8600.

*Trademark of Digital Research

S-100 PRINTER INTERFACE

MicroPro has introduced I/OMaster, a new S-100 printer interface which adds 30% or more printer speed performance while reducing user costs by allowing flexible use of either lower cost letter-quality printers and/or high speed line printers within the same microcomputer configuration. I/OMaster combines four boards in one, featuring two each serial and parallel ports as well as an eight-level interrupt control and dual interval timer circuitry. All I/OMaster options are DIP switch selectable.

Available from over 350 dealers across the U.S. and overseas, I/OMaster costs $400 complete and tested with factory warrantee.

For more information MicroPro International Corporation, 1299 Fourth Street, San Rafael, CA 94901, (415) 457-8990.

In each issue of S-100 MICROSYSTEMS we will have this catalog listing of S-100 system software. If you have a software package you are offering for sale and want to be listed then send us the information in the format shown. All information must be included. We reserve the right to edit and/or reject any submission.

SOFTWARE DIRECTORY

Program Name: TED
Hardware System: 24K or larger Z80 CP/M system
Minimum Memory Size: 20K minimum, 24K recommended
Language: Z80 assembly
Description: TED is an advanced text editor which implements an enhanced subset of DEC TECO commands providing the following capabilities for editing of ASCII text.

*36 command/text buffers
*32 entry push down stack
*sophisticated macro command capability
*conditional and iterative command execution
*conditional and absolute branching
* multiple open files

Release: Available now
Price: $90.00
Included with price: 8" CPM compatible disk with object file, TED. COM and comprehensive manual (manual $20 if purchased separate).
Where to purchase it:
Small System Design
P.O. Box 4546
Manchester, New Hampshire 03108

Program Name: MWP 2.0-MINI WORD PROCESSING
Hardware System: CP/M
Minimum Memory Size: 48K bytes
Language: Microsoft BASIC
Description: Mini Word Processing enables the user to prepare letters, text, mailing labels and envelopes. Information stored in user defined name and address files can be inserted throughout a letter or text. Documents can be assembled from any number of files stored on the disk, and can be printed or displayed on the CRT with user selectable margins, page size, headers, page numbers and insertions.
Release: Available now
Price: $195 Licence Agreement Required
Included with price: Disk, Manual, examples, support
Author: The Software Store
Where to purchase it:
THE SOFTWARE STORE
706 Chippewa Square
Marquette, MI 49855

Program Name: MULTI-USER CP/M
Hardware System: CP/M with 8" floppy disk
Minimum Memory Size: 48K bank switchable memory
Language: 8080 machine code
Description: Allows up to four terminals to be supported from one 8080/Z-80 computer. Will also support up to four printers. Lock-out is provided for printer use. System requires one copy of CP/M 1.4 and an interrupt board to generate restart 6 every 16-20 MS.. DMA type disk controller is recommended for faster system performance.
Release: Available now
Price: $125.00 List
Included with price: Diskette and Manual
Author: Mark Winkler
Where to purchase it:
Provar, Inc.
6217 Kennedy Avenue
Hammond, Indiana 46323

Program Name: RF
Hardware System: North Star MDS or Horizon
Minimum Memory Size: 8K Bytes
Language: Assembler, distributed as object code
Description: A file renaming utility designed in the spirit of the disk utilities now supplied with North Star. With RF a simple one-line command is sufficient to change a file name; e.g. the command "GO RF ABC, 2 XYZ" changes file ABC on drive 2 to XYZ. RF prompts for missing parameters and generates meaningful error messages, i.e. "NEW-NAME IN USE." File renaming is a high activity function and should not be left to clumsy, error-prone methods.
Release: Currently available
Price: $7.95 ($1.00 documentation only, credited to purchase); check or money order
Included with price: Disk, documentation, postage & handling; specify single or double density
Author: Jim Hendrix
Where to purchase it:
Jim Hendrix
Rt. 1, Box 74-B-1
Oxford, MS 38655

Program Name: Small-C Compiler
Hardware System: 8080/Z80
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAC as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program inc.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Speedy Disk Copy
Hardware System: Mits Altair with 88-DCDD disk controller & two 8" hard sectored disk drives.
Minimum Memory Size: 32K RAM
Language: Altair Disk Basic (Rev. 3.4, 4.0, 4.1, & 5.0)
Description: The Speedy Disk Copy routine will copy all or any part of an Altair Disk Basic or DOS formatted diskette in less than 100 seconds. By comparison, the Altair supplied PIP utility requires about 40 minutes. The program is self-prompting, performs both read and write validation checks with a listing of the location and quantity of errors upon completion. Recorded twice in Altair Disk Basic ASCII format on 8" hard sectored diskette.
Release: Currently
Price: $19.50 Postpaid.
Author: Joe Konrad
Where to purchase it:
Jack Compute
33 Plant Street
New London, CT 06320

THE ONLY MAGAZINE BY AND FOR S-100 SYSTEM USERS!

S-100 MICROSYSTEMS™

At last there is a magazine written exclusively for S-100 system users. No other publication is devoted to supporting S-100 system users. No longer will you have to hunt through other magazines for an occasional S-100, CP/M* or PASCAL article. Now find it all in one publication. Find it in **S-100 MICROSYSTEMS.**

Every issue of **S-100 MICROSYSTEMS** brings you the latest in the S-100 world. **Articles on applications, tutorials, software development, letters to the editor, newsletter columns, book reviews, new products, etc.** Material to keep you on top of the ever changing microcomputer scene.

SOFTWARE
CP/M*
Assembler
BASIC
PASCAL
applications
and lots more

SYSTEMS
Cromemco
Intersystems
North Star
IMSAI
SOL
and lots more

HARDWARE
8 bit & 16 bit CPUs
interfacing
hardware mods
bulletin board systems
multiprocessors
and lots more

*TMK Digital Research

Edited by Sol Libes
Published every other month

USA	Canada, Mexico	Other Foreign (Air)
	THREE YEARS (18 issues)	
☐ $23.00	☐ $32.00	☐ $65.00
	TWO YEARS (12 issues)	
☐ $17.00	☐ $23.00	☐ $45.00
	ONE YEAR (6 issues)	
☐ $9.50	☐ $12.50	☐ $23.50

☐ New ☐ Renewal
☐ Payment Enclosed
☐ Visa
☐ MasterCard
☐ American Express
Signature _____
Card No. _____
Expiration date _____

☐ Please bill me ($1.00 billing fee will be added; foreign orders must be prepaid)

S-100 MICROSYSTEMS
P.O. Box 789-M, Morristown, NJ 07960

S-100 MICROSYSTEMS™ ORDER FORM

Name _____

Address _____

_____

City _____ State _____ Zip _____

BACK ISSUES

☐ 1-1 Jan/Feb 1980 $5.00
☐ 1-2 Mar/Apr 1980 $2.50
☐ 1-3 May/Jun 1980 $2.50
☐ 1-4 Jul/Aug 1980 $2.50
☐ 1-5 Sep/Oct 1980 $2.50
☐ 1-6 Nov/Dec 1980 $2.50

Postpaid in USA; add $1.00 per issue foreign postage. Subscriptions start the month following receipt of order. Subscriptions cannot start with earlier issues.

Program Name: Master Ledger
Hardware system : CP/M, 48K, 2-8" Drives
Language: CBASIC-2
Description: Master ledger analyzes your business. It includes 12 month's budgets plus 12 month account history, making possible for any tape of financial comparisons will show management if they are meeting their financial goals. Quarterly and yera-to-date are also available for any period. Ten different journals, general ledger, trial balance, and budgets are some of the other major reports. The input routines were designed for the operator, easy, fast and verifies all input. Special features include a forced audit trail and a forced balancing system.
Release: Available now
Price: $800.00
Included with price: User documentation, 31 computer programs, warranty
Author: Keystone systems, Inc.
Where to purchase it:
Keystone Systems, Inc.
P.O. Box 767
Spokane, WA 99210.

Program Name: MCALL (Micro proto'CALL' & communications program)
Hardware System: 8080 or Z80 under CP/M serial port connected to an acoustic coupler, video display terminal.
Minimum Memory Size: 16K
Description: MCALL provides the following major functions:

1. Time Sharing Terminal emulation.
2. Disk file transfer between PC (Personal Computer) and TSC (Time Sharing Computer) in either direction.
3. Disk file transfer between two PC's with error detection and retransmission.

To perform function 3, no coordination between operators is required. The file to be transferred is specified by the transmitting operator; then the transmit command is issued (ESC T). The specified file is automatically opened at the TX end and created at the RX end. Subsequently, each file is closed and a message notifies each operator that the transfer was a success (or not) and displays the total retransmission count. The INFOWORLD Software Report Card for MCALL rated: "Ease of Use" and "Support" as excellent; and "Functionality", "Documentation", and "Error Handling" as good.
Release: Currently available
Price: $50.00
Included with price: Operating Instructions Manual (20 pgs) and 8" single density disk with 84K source file and 8K com file.
Author: Tim Pugh
Where to purchase it:
Micro-Call Services
9655-M Homestead Ct.
Laurel, MD 20810

COMPUTERIZED POST OFFICE

USEFUL IN REMOTE MODEM OR MULTI-USER COMPUTER SYSTEMS.

FEATURES ----------

1: BOX CONFIGURATION ---- 100 BOXES WITH 10 MESSAGES PER BOX.
2: SECURITY ---- EACH BOX IS KEYWORD PROTECTED. MESSAGE FILE CAN NOT BE SEEN BY SYSTEM USERS.
3: LINE EDITOR ---- 7 CONTROL KEYS RECOGNIZED (CTL-H, CTL-X, CTL-R, ETC) OTHER FEATURES INCLUDE STRING SUBSTITUTION AND DELETE, CONTINUE, TYPE LINE, TYPE MESSAGE, REPLACE LINE.
4: BULLETIN LINE AND BOX ---- NOTICES AND BULLETINS ACCESSIBLE BY ALL.
5: DYNAMIC FILE ALLOCATION ---- NO FILE CRUNCHING NECESSARY. DELETED FILE SPACE IS RELEASED BACK TO THE PROGRAM.
6: SPECIAL OPERATOR COMMANDS ---- BOXES CAN BE ACTIVATED OR DEACTIVATED. BULLETIN LINE AND BOX CAN BE UPDATED.

PRICE $50.00 FOR COM FILE OR $100.00 WITH SOURCE WRITTEN IN PASCAL/MT. FOR CPM 2.X OR MPM 1.X.

OTHER SOFTWARE PACKAGES

SPACE INVADERS ---- REAL TIME, NO PAUSES DURING EXPLOSIONS AND ETC. VDM TYPE DISPLAYS $25.00 WITH SOURCE IN PASCAL/MT.

FIXFOR ---- CONVERTS SINGLE DENSITY DISKS FOR USE WITH TARBELLS NEW DOUBLE DENSITY CONTROLLER. $25.00 WITH 8080 SOURCE.

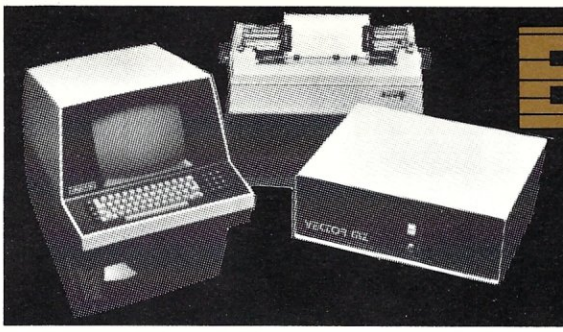
XIOS.ASM ---- MPM BIOS FOR TARBELLS DOUBLE DENSITY CONTROLLER. $45.00

MPM UTILITY PACKAGE ---- PSPOOL, PRINT, LOGIN, DR, MESSAGE, BPATCH, BSHORT, AND ASHORT. $35.00 WRITE FOR DETAILS.

SUPPLIED ON AN 8 INCH SINGLE DENSITY DISK

MARK WINKLER
541 INGRAHAM AVE.
CALUMET CITY, ILLINIOS
60409
312-868-4866

ADVERTISER INDEX



Budget InfoSystems

25% Down - Balance C.O.D. - Prepaid orders save freight charges. Minnesota residents must add 4% sales tax or provide evidence of tax number for items to be resold. Twin city - metropolitan prices are slightly higher.

CRT TERMINALS:	RETAIL	BIS PRICE (Qty. one)
All ADDS Regent Terminals in stock (20, 25, & 40)		10% OFF
Microterm 2A	995.00	850.00
Microterm 5A	945.00	795.00
Microterm 314	895.00	750.00
Microterm 100 (DEC VT100 emulator)	1895.00	1695.00
TVI 912B & 912C	to 895.00	795.00
TVI 920B & 920C	to 995.00	850.00
MATRIX PRINTERS (RO):		
Anadex 9500 printer	1650.00	1475.00
Centronics 779TF	1350.00	1095.00
Centronics 737 (add $95 for serial)	995.00	850.00
DAISYWHEEL CHARACTER PRINTERS (RO):		
BudgetRiter III (or Qume 3/55) w/S100 interface card	3095.00	2495.00
Qume Sprint 5/55	3195.00	2895.00
Plessy CP4282 (by Data Products)	3595.00	2595.00
MASS STORAGE SUBSYSTEMS:		
BudgetStor I-dual Micropolis MOD IIs (630K)		1595.00
BudgetStor II-dual 8" drives (1.2MB)		1495.00
BudgetStor III-dual 8" Qume drives (2MB)		1895.00
Micropolis 1053 MOD II 2 drive system	1895.00	1695.00
KONAN "Hard Tape" controller, with Century Data 10MB disk & DEI 15.5MB tape drive	6455.00	5995.00
Same with 20MB disk drive	6915.00	6395.00
Same with 40MB disk drive	7685.00	6995.00
MODULAR S100 MICROCOMPUTERS (MAINFRAMES ONLY):		
Watch for the introduction of our own "BudgetSystem 696" - 64K RAM, dual Qume 8" drives (2MB), "UNIX-like" DOS, and more! Currently in its final stages of development. Available options will include: 45MB 8" hard disk, Z8000 CPU, PROM "burner", multi-terminal OS, and a wealth of software!		
Intersystems DPS-1(w/front panel)	1795.00	1625.00
Intersystems DPS-1FPL (front Paneless)	1395.00	1275.00
Intersystems System 1A (w/64K, Z80 & I/O)	3195.00	2875.00
Intersystems System 2A (same plus FDC2)	3595.00	3250.00
Intersystems System DPX-1 (1A w/Z8001)	4895.00	4425.00
Intersystems System DPX-2 (2A w/Z8002) (other configurations available on request)	4795.00	4325.00
Vector Graphic MZ 64K, CP/M2, etc.	4313.00	3495.00
VECTOR GRAPHIC S100 SYSTEMS:		
Basic System B (64K RAM, 630K disk)	5463.00	4395.00
Basic System 2800 (64K RAM, 2MB disk)	7295.00	5895.00
Basic System 3030 (32MB Hard Disk)	11945.00	9595.00
(All Vector systems available with accounting, word processing, and specialty software packages at VERY reasonable prices!)		
S100 BUS CIRCUIT CARD ASSEMBLIES (ASSEMBLED & TESTED):		
Intersystems DPS Front Panel card	495.00	445.00
Intersystems MPU-8000-1 CPU card	1395.00	1255.00
Intersystems MPU-8000-2 CPU card	1245.00	1120.00
Intersystems MPU-80 card	395.00	355.00
Intersystems 16KSR 16K Static (250ns)	495.00	445.00
Intersystems 64KDR 64K Dyn. RAM card	995.00	895.00
Intersystems FDC2 DD DMA disk controller	495.00	445.00
Intersystems VIO-1 (4P + 2S) I/O card	445.00	400.00
VIO-O Same without interrupts	395.00	355.00
Intersystems 6SIO Serial I/O card	895.00	805.00
Intersystems EP64 64K EPROM card	295.00	265.00
Intersystems EPB-M EPROM programmer	395.00	355.00
Intersystems ADDA 8-bit A/D & D/A card	495.00	445.00

S100 BUS CIRCUIT CARD ASSEMBLIES	RETAIL	BIS PRICE (Qty. one)
Intersystems WW1 Prototyping card	NET	35.00
Konan SMC-100 Hard Disk Controller	2195.00	1795.00
NEW! Vector Graphics "ZPB" SBC card	395.00	325.00
Vector Z80A CPU card	247.00	195.00
Vector Bitstreamer II card	270.00	225.00
Vector Motherboard, 18 slot, terminated	201.00	160.00
Vector Analog Interface Card	132.00	110.00
Vector Precision Analog Interface Card	449.00	350.00
Vector High Res. Graphics Card	270.00	225.00
Vector Video Digitizer Card	201.00	160.00
Vector Flashwriter II 80 x 24 Video	368.00	295.00
Vector PROM/RAM III Card	247.00	195.00
Vector 8K Static RAM Card	282.00	195.00
Vector 64K dyn. RAM Card	975.00	795.00
Micropolis Controller Card	518.00	425.00
CF&A DATA DESKS:		
Universal Printer Stand (all metal)	130.00	115.00
CRT data desk 48" wide	178.00	160.00
data desk return 18" x 36"	145.00	135.00
"Microshelf" (specify mainframe size)	110.00	95.00
"Microshelf" vented for Vector MZ	135.00	120.00

28" Ht. and 30" depth are standard; returns are 26" ht.
Top colors: Teak, Walnut, Pecan, Black, White, Putty, Champagne
Nonstandard tops are $25.00 extra (we prefer Rosewood or leathers)
Sheetmetal colors: Champagne, Black, Blue, Yellow, Putty, Brown, Butterscotch; legs are black.
The complete line of CF&A RETMA (19") racks, desk options, fill-in panels, etc. are also available - call for prices.

MISCELLANY:		
Anderson-Jacobson A242A (remanufactured)	185.00	150.00
Novation CAT ansr/orig. modem	189.95	175.00
Qume Single sheet paper feeder	1390.00	1195.00

OPUS SD/SS Mini-diskettes 10 in plastic box — specify 0, 10, or 16 sectors & system SPECIAL 27.50/10 250.00/100 in bulk

OPUS SD/SS 8" Diskettes 10 in plastic box — specify 0 or 32 sectors & system SPECIAL 29.50/10 265.00/100 in bulk

OPUS RECORDING MEDIA ARE A PREMIUM GRADE FULLY GUARANTEED PRODUCT OFFERING SUPERIOR VALUE...DOUBLE DENSITY OR SPECIAL FORMATS AVAILABLE AT SLIGHTLY HIGHER PRICES.

CompuClean head cleaning diskette 5¼" 19.00
CompuClean 8" 19.00
(Specify single or double sided) three packs only 47.50
These are a dry process - no mess & much safer!

SOFTWARE:

OS-1 "UNIX-like" Z80 Optimized Operating System & Utilities (fully CP/M compatible) $250.00

We have the complete line of Microsoft software available - all prices 10% off.

MDBS Data Base Management System (CODASYL-compatible) the last word in DBMS's! available in a wide variety of formats, with or w/o optional modules from $750.00
HDBS Data Base Management System - the 2nd to the last word!!! $350.00

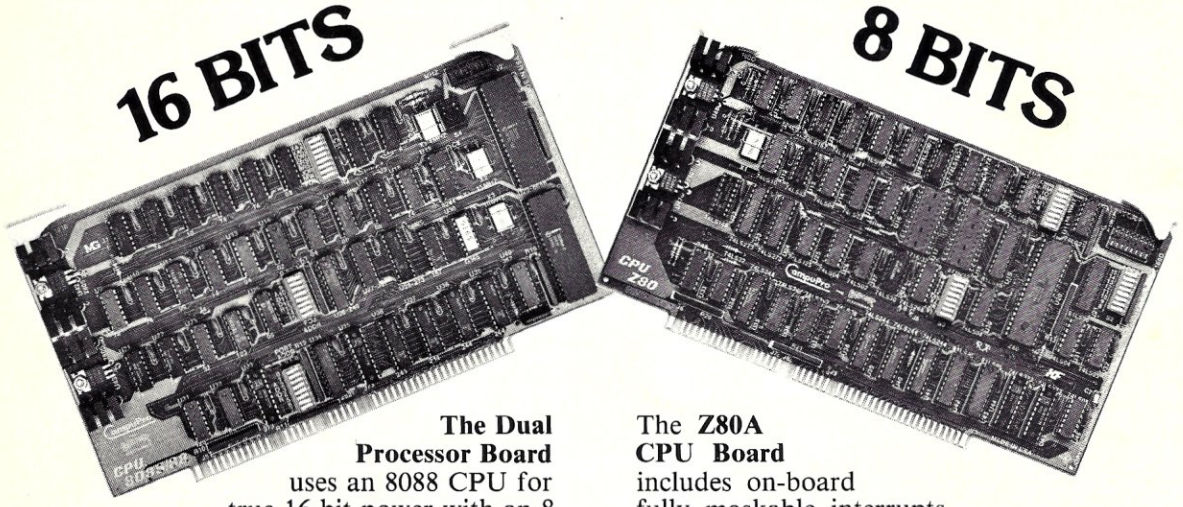
DEVELOPMENT SYSTEMS & APPLICATIONS PROGRAMS

We have a wide variety of development and applications packages from several of the major software houses (RSI "Peachtree", IMS, Graham-Dorian, and others), plus our own line of proprietary software packages. All attractively priced.

BUDGET INFOSYSTEMS
1633 N.E. Highway Ten
Spring Lake Park, MN 55432
(612) 786-5545
Normal Order Line Hours Are: 12:00 to 6:00 p.m. CST

Throughput is the Only True Measure of Computer Performance.

Want a 300% improvement in throughput compared to 2 MHz systems? IEEE-compatible CompuPro boards are designed *from the ground up* to operate at 6 MHz and beyond, dramatically increasing computing power and performance. Don't settle for less . . . select high speed, high reliability S-100 products from CompuPro.



The Dual Processor Board uses an 8088 CPU for true 16 bit power with an 8 bit bus, and an 8085 for compatibility with CP/M and 8080 software. **SPECIAL LOW PRICES: $295 unkit, $425 assm** (both operate at 5 MHz); **$525** qualified under the high - reliability **C**ertified **S**ystem **C**omponent program (with 5 MHz 8085, 6 MHz 8088).

The **Z80A CPU Board** includes on-board fully maskable interrupts for interrupt- driven systems, provision for adding up to 8K of on-board EPROM, IEEE compatible 16/24 bit extended addressing, and much more. 4 MHz standard operation, but also works with 6 MHz Z80s. **$225 unkit, $295 assm $395 CSC.**

HIGH SPEED S-100 MEMORY and MOTHERBOARDS

RAM XX (with bank select AND extended addressing) is the perfect match for either CPU board — thanks to fully static operation, extremely low power consumption, and complete IEEE spec compatibility. All **unkit** and **assembled** memories work up to 5 MHz, while **C**ertified **S**ystem **C**omponent boards run up to 8 MHz and are guaranteed to work with 6 MHz Z80s. All CompuPro motherboards work up to 10 MHz.

	unkit	assm	CSC
16K RAM XX-16	**$349**	**$419**	**$519**
24K RAM XX-24	**$479**	**$539**	**$649**
32K RAM XX-32	**$649**	**$729**	**$849**
20 slot motherboard with edge connectors	**$174**	**$214**	n/a
12 slot motherboard with edge connectors	**$129**	**$169**	n/a
6 slot motherboard with edge connectors	**$89**	**$129**	n/a

SEE COMPUPRO PRODUCTS IN PERSON AT COMPUTER STORES WORLDWIDE, OR WRITE US DIRECT IF THERE'S NO STORE IN YOUR AREA.

TERMS: Cal res add tax. Allow 5% for shipping, excess refunded. VISA®/Mastercard® orders call **(415) 562-0636**, 24 hours. Please include street address for UPS delivery. Sale prices good through cover month of magazine, other prices subject to change without notice.

CompuPro™ from GODBOUT ELECTRONICS

Bldg. 725, Oakland Airport, CA 94614